University of Sussex

Data-Driven Techniques for Animating Virtual Characters

Christos Mousas

Submitted for the degree of Doctor of Philosophy

University of Sussex

October 2014

UNIVERSITY OF SUSSEX

Christos Mousas

Data-Driven Techniques for Animating Virtual Characters

Summary

One of the key goals of current research in data-driven computer animation is the synthesis of new motion sequences from existing motion data. This thesis presents three novel techniques for synthesising the motion of a virtual character from existing motion data and develops a framework of solutions to key character animation problems.

The first motion synthesis technique presented is based on the character's locomotion composition process. This technique examines the ability of synthesising a variety of character's locomotion behaviours while easily specified constraints (footprints) are placed in the three-dimensional space. This is achieved by analysing existing motion data, and by assigning the locomotion behaviour transition process to transition graphs that are responsible for providing information about this process.

However, virtual characters should also be able to animate according to different style variations. Therefore, a second technique to synthesise real-time style variations of character's motion. A novel technique is developed that uses correlation between two different motion styles, and by assigning the motion synthesis process to a parameterised maximum a posteriori (MAP) framework retrieves the desire style content of the input motion in real-time, enhancing the realism of the new synthesised motion sequence.

The third technique presents the ability to synthesise the motion of the character's fingers either off-line or in real-time during the performance capture process. The advantage of both techniques is their ability to assign the motion searching process to motion features. The presented technique is able to estimate and synthesise a valid motion of the character's fingers, enhancing the realism of the input motion.

To conclude, this thesis demonstrates that these three novel techniques combine in to a framework that enables the realistic synthesis of virtual character movements, eliminating the post processing, as well as enabling fast synthesis of the required motion.

Acknowledgements

This work could not be accomplished without support from many people.

First of all, I am extremely thankful to my supervisors Dr. Paul Newbury and Dr. Martin White for their valuable support, and their inexhaustible patience that he has shown during the elaboration of this thesis. Throughout this work they encouraged me to pursue my own research interests and to develop independent thinking and research skills.

I need to thank Dr. Christos-Nikolaos Anagnostopoulos (Associate Professor, University of the Aegean) for his unofficial supervision during my time at the University of the Aegean, as well as for the financial support that he provided to me through the Digital Herodotus (European Territorial Cooperation Cyprus - Greece) research grand.

I would like also to thank my best friends I. Andreopoulos, C. Ouzounis, A.-F. Kilias, A. Petridis, D. Kotsilinis, N. Grigoriou, A. Bountonas, and E. Hatzikiriakou for their honest and moral support through this challenging journey.

Finally, I would like to thank my parents Efstratios and Efthimia, and my sister Maria, for always believing and unconditionally supporting me. I consider myself very lucky to have them in my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Virtual humans as well as virtual worlds can be described as virtual artefacts. In general, those artefacts are imaginary objects that exist in the human mind. During the past years, the computer graphics community has developed various technologies providing the ability to give life to the human imagination. Virtual characters and virtual worlds can now be designed and produced by those experienced with the tools.

These days, computer animated characters can be encountered in various applications related with virtual reality, such as in video games, virtual worlds, films etc. Pausch et al. (1996) Steuer (1992). The need to represent real life with virtual artefacts is required in order to enhance the imagination of the observers. Hence, computer animated characters generated by humans imaginations in the near future are not going to replace the actors, especially in the film industry. They are coming to represent either realistically Shiratori et al. (2009) Wu and Zordan (2010) or in a cartoonistic way Wang et al. (2006) Chenney et al. (2002) the human's activities in a virtual world or to enhance the human abilities Yamane et al. (2010) Yamane and Sok (2010) such as by synthesising supernatural complex motions.

In this chapter, firstly, it is introduced the structure of the articulated human body, as well as its representation in the virtual space. Then, it is introduced the motion capture as the ability of animating virtual characters. Continuing, the pipeline of the contemporary computer animation, as well as the ability of inversing the content creation is represented. Additionally, it is analysed the presented research and the way that some of the basic problems approached and solved. Moreover, the contribution of the presented research, as well as the research outcomes that were covered during the PhD studies, are briefly described. Finally, the structure of the thesis is presented in the last subsection.

1.1 The Articulated Virtual Human's Body

The structure of human body is quite complex. In general, its structure consists of head, neck, torso, two arms, and two legs. By the time where a human reaches its adulthood, the body consists of close to 100 trillion cells Greg (2007) that is the basic unit of life Alberts (2007). These cells are organised biologically to eventually form the whole body of the human. When virtual characters are coming to mimic the movements of real humans, the need of modelling and simulating each individual part of the human body was raised. Although, in contemporary computer animation research, there have not been examined and developed any fully functional virtual human's body yet. This is since simulating such a complex system requires not only computer graphics and computer animation related knowledge, but it is also required the knowledge of how each individual part of the human's system works. Hence, the researchers focus on specific parts of the human's body for simulating its activities.

Among other parts, human's body consists of the musculoskeletal system. Since the presented research focuses on human motion synthesis, following it is presented the articulated body model of the humans (see Figure 1.1). More specifically, human's skeletal model is a highly complex hierarchical model (see Figure 1.2) that consists of many joints, where each one has different degrees of freedom and various possible restrictions. In fact, the human's body consists of more that 200 bones and joints.

The virtual human's body can be described as a multibody system that consists of a set of rigid objects, generally called links that are connected together by a joint. In general, a joint is a component that it is concerned with motion, and it permits some degree of relative motion between the connected segments. In computer animation, the ability of modelling the virtual body is quite important, especially for controlling the human posture. Hence, while it is desired a motion that will be as natural as possible, a well-constrained model restricts postures to a feasible set. On the other hand, it is assumed that the body parts are rigid, although this is just an assumption that approximates the reality. In general, the skeletal structure is usually modelled as a simplified hierarchy of rigid segments that are connected by joints, where each one is defined by its length, shape, volume, and mass properties. Concluding, the skeletal structures are often defined by using a parent-children system. Hence, the size, the shape, and other properties of the body and its segments are also essential in order to build models with realistic dimensions and properties. In Figure 1.3 the virtual skeleton of a character is illustrated.

Figure 1.1: Human's articulated body model Plasticboy (2014).

1.2 Computer Animation via Motion Capture

Motion capture (MoCap) is the ability to capture the 3D live action in real-time either the whole-body or specific parts of human body like face, hands etc. MoCap works by tracking motion of reference points and then convert these reference points to joint angles to drive an articulated 3D model or to drive a deformable surface. MoCap system captures the X, Y, Z position as well as the roll, pitch and yaw. In general, the motion data retrieved from the motion capture devices can be used either off-line (choose among them, blend between them, modify on the fly etc. for synthesising new animated sequences), or on-line (performance animation) while driving characters directly based on the actor's performance in real time. A general pipeline of the motion capture process is illustrated in Figure 1.4. Specifically, for capturing the human motion the MoCap systems requires the following procedure. Firstly, a calibration process of the attached to the human markers is required, such as enabling the MoCap system to understand the position of each marker. Secondly, the system captures the markers and estimates the skeleton of the human. Finally, inverse kinematics solutions are coming to animate the character's joints according to the human's performance.

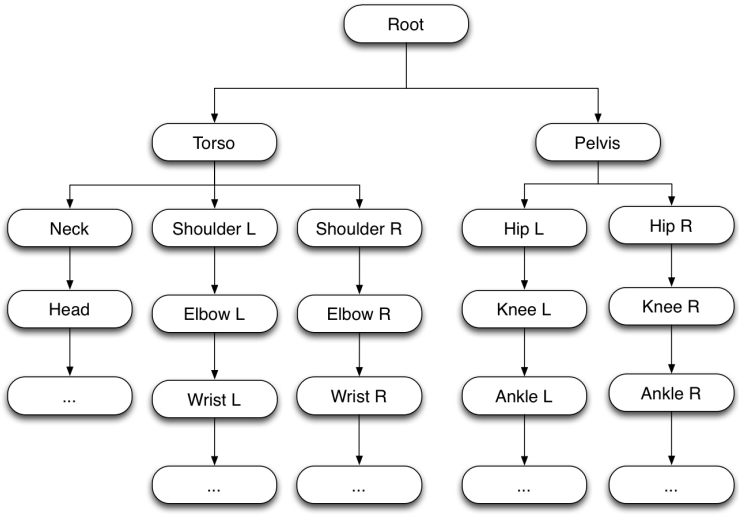

Figure 1.2: The hierarchical graph of skeleton joints.

1.2.1 Human Motion Capture Technologies

The technologies for capturing the human motion can be separated in two major categories namely the optical and the non-optical, as those presented in the following subsections.

Optical Motion Capture

In general, optical motion capture systems work with one or more cameras. Usually the subject is equipped with reflective patches or spheres (called markers), indicating the position to the cameras. The markers themselves are tracked using software solutions. By combining the same markers on multiple cameras (which all have a different position), a 3D position of a marker can be determined. The cameras are often infrared and mounted to a rig or to the walls of a room Magnenat-Thalmann and Thalmann (2005). A simple example of a performer wearing the require markers is illustrated in Figure 1.5.

During the last years, methodologies for approximating the human's shape, gave the ability of developing markerless optical motion capture systems, which are mainly based on computer vision methodologies. Hence, several commercial solutions for markerless MoCap have been introduced, including systems by Organic Motion Organic Motion (2014) and Xsens Xsens (2014). The ability of capturing human's motion without the requirement of markers it is based on computer vision algorithms that allows the system to analyse multiple streams of optical input, identifying the human forms, breaking them down into constituent parts for tracking. Since the basic limitation of those solution is the high cost, hardware such as the Microsoft's Kinect Microsoft (2014) and Assus Xtion Assus (2014) provides the ability of capturing the desired motion sequences.

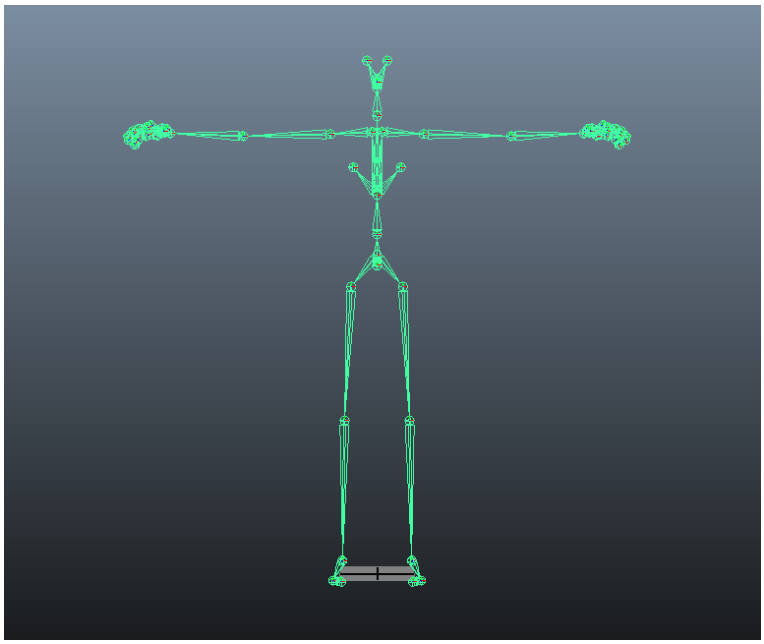

Figure 1.3: A virtual skeletal structure that describes the real human's skeletal system in computer animation.

Non-Optical Motion Capture

The non-optical motion capture systems are divided in three major categories namely, the inertial, the mechanical, and the magnetic MoCap systems. More specifically, the inertial MoCap technology consists of inertial sensors, biomechanical models and sensor fusion algorithms. The motion data of the inertial sensors is often transmitted wirelessly to a computer, where the motion is recorded or viewed. Most inertial systems use gyroscopes to measure rotational rates Vlasic et al. (2007) Miller et al. (2004). These rotations are translated to a skeleton in the software. On the other hand, the mechanical MoCap systems, often referred to as exo-skeleton motion capture systems due to the way the sensors are attached to the body, are responsible for directly tracking the body joint angles. Performers attach the skeletal-like structure to their body and as they move so do the articulated mechanical parts, measuring the performer's relative motion. Finally, magnetic systems calculate position and orientation by the relative magnetic flux of three orthogonal coils on both the transmitter and each receiver. The relative intensity of the voltage or current of the three coils allows these systems to calculate both range and orientation by meticulously mapping the tracking volume. The sensor output is 6 degrees of freedom (DOF), which provides useful results obtained with two-thirds of the number of markers required in optical systems; one on upper arm and one on lower arm for elbow position and angle. In Figure 1.6 those three types of deferent non-optical motion capture

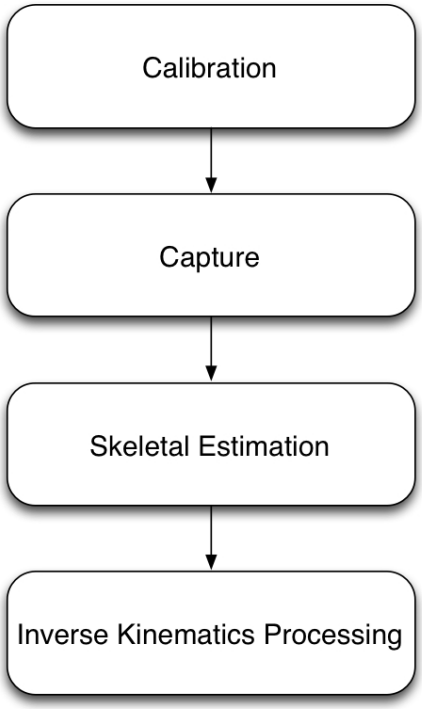

Figure 1.4: The motion capture process pipeline.

systems are illustrated.

1.2.2 Motion Capture Databases

The usage of MoCap data either for analysing or for synthesising the human motion attracted the research and educational community. Various research institutes and University laboratories provide their own MoCap datasets for research reasons MPI Informatik (2014). Hence, since the research community can use the same datasets it is possible to analyse in a more consistent fashion each research outcome.

In this research, the use of MoCap data plays a crucial role. This is especially true, while it is desired a direct comparison of the presented methodologies with previously presented solutions. This is since the motion synthesis process can be evaluated against the effectiveness of the methodology to estimate and synthesise the required motion, while the error of the motion synthesis is computed. By using existing motion data that has been used in previous research, it is possible to evaluate the computed error against the results from any other solution. Therefore, it is possible to be shown the advantages that are provided by new methodologies.

Figure 1.5: The performer wears the necessary markers that are required for the motion capture process (Figure from Savannah College (2014)).

1.2.3 Capturing, Representing, and Correcting Human Motion Data

During the past decades various methodologies for capturing the human motion have developed. Although, since the MoCap data always increases, it is required to be established a general representation of human motion data. The software developing companies especially requires an efficient representation of MoCap data. This is since, they are calling to provide software solutions for being able everyone to work with the motion capture file formats. Additionally, the ability of establishing standard file formats for the MoCap data is required as well by the research community, being able everyone to use in an efficient way those motion files. Hence, among the most widely used motion file formats are the BioVision Hierarchical Data (BVH), the Acclaim Skeleton File/Acclaim Motion Capture data (ASF/AMC), and the C3D file format. For an extended explanation of the aforementioned file formats, it is referred to read Meredith and Maddock (2001).

The BVH File Format

Since in this research the BVH motion data was used, following section explains its representation. It should be noted that it was chosen to work with the BVH file format since it provides the ability of handling in an easy and efficient way the motion data. Moreover,

Figure 1.6: Three non-optical motion capture systems. The Inertial (left), the Mechanical (middle), and the Magnetic (right) motion capture systems (Figures from Metamotion (2014)).

the low storage of information included in a BVH file (a motion sequence of 10 seconds is approximated in 30KB) makes this format quite usable especially in large-scale applications where a huge number of motion data is required to be used. Such a file format can be recognised in various software packages, such as in Autodesk's Maya Autodesk (2014a) and Motionbuilder Autodesk (2014b), in Blender Blender (2014) etc. making its quite useful while it is required a cross platform communication.

A BVH file contains ASCII text, the first part of which provides the specifications for the initial pose of a human skeleton, and the rest a time-framed sequence of different specifications for subsequent poses. Headed by the keyword "HIERARCHY", the first part of a BVH file identifies the Hips joint as the "ROOT", which is to say it has no parent-joint. This is the starting point of a nested-structure of parent-joints and child-joints. An example for this hierarchy is illustrated in Figure 1.7.

The "ROOT" section specifies the location of the Hips joint in a three-dimensional space. In Figure 1.8 the "ROOT" section in the structure are "JOINT" sections. Each section contains information that specifies the location of the skeletal joint according to the parent joint. Generally, the relative location between parent and child joints makes it possible to work out of the length of the bone between two joints. It should be noted that there are joints that has not a child joint, which are generally called as end-effectors. The "ROOT" section and each of the "JOINT" sections of a BHV file specifies the "CHANNELS". Each of which processes the time-framed sequence of translation and/or rotational

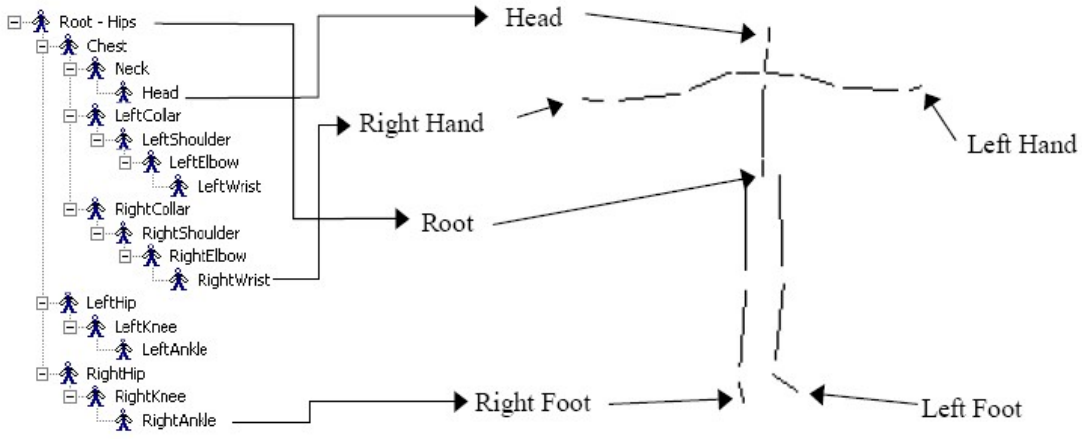

Figure 1.7: BVH skeletal structure.

co-ordinates provided in the second part of the BVH file. It should be noted that root joint channel takes six parameters which are the X, Y, Z positions and the X, Y, Z orientation. Conversely, any other joint takes only three channels, which are the X, Y, Z orientation.

```
HIERARCHY
ROOT hip
{
  OFFSET 0 0 0
  CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
  JOINT abdomen
  {
    OFFSET 0 20.6881 -0.73152
    CHANNELS 3 Zrotation Xrotation Yrotation
    JOINT chest
    {
      OFFSET 0 11.7043 -0.48768
      CHANNELS 3 Zrotation Xrotation Yrotation
      JOINT neck
      {
        OFFSET 0 22.1894 -2.19456
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT head
        {
          OFFSET -0.24384 7.07133 1.2192
          CHANNELS 3 Zrotation Xrotation Yrotation
          JOINT leftEye
          {
            OFFSET 4.14528 8.04674 8.04672
            CHANNELS 3 Zrotation Xrotation Yrotation
            End Site
            {
              OFFSET 1 0 0
            }
          }
          JOINT rightEye
          {
            OFFSET -3.6576 8.04674 8.04672
            CHANNELS 3 Zrotation Xrotation Yrotation
            End Site
            {
              OFFSET 1 0 0
            }
          }
        }
      }
    }
```

Figure 1.8: The first half of a BVH file.

The second part of a BVH file (see Figure 1.9) starts with the keyword “MOTION”, followed by information specifying, first, the number of FRAMES; second, the sampling rate per second after the keyword “FRAME TIME”; and third, a number of lines corresponding to the number of “FRAMES”, each line providing the translation and/or rotation coordinates to be processed according to the “CHANNELS” specifications in the first part

of the file.

```
MOTION
Frames: 3303
Frame Time:     0.00833333
-4.82154 84.8033 31.0821 0.0 0.0 0.0 -0.305086 -0.00406093 -0.0087977 -0.0843762 -0.00666545 -0.0146211 -3.88453e-15
7.95139e-16 -9.93923e-17 -1.17656e-14 3.18055e-15 -2.95071e-16 -0.0 0.0 0.0 -0.0 0.0 0.0 -1.84021e-15 -1.12593e-17
9.93923e-17 -3.94017e-14 9.93923e-17 -1.98785e-16 -7.05387e-13 2.06935e-13 -1.59028e-14 -8.09844e-12 -2.48456e-13
-2.13097e-13 -1.91022e-11 -9.76324e-14 -4.80264e-13 -3.29998e-11 -1.74931e-13 -8.39666e-13 -1.9098e-11 -9.81996e-14
-4.92986e-13 7.73091 5.47488 0.417314 -1.91035e-11 -9.79015e-14 -4.83444e-13 2.90554 0.965183 0.102902 -1.90955e-11
-9.77027e-14 -4.82649e-13 10.2786 0.731801 0.0259123 -1.91104e-11 -9.46215e-14 -4.80264e-13 2.22167 -0.733487 -0.20892
-2.03622e-15 -1.10652e-17 -1.98785e-16 -4.21245e-14 1.05604e-16 -2.18663e-15 -7.10755e-13 -2.17669e-13 6.36111e-15
-8.10822e-12 1.78509e-13 -2.25819e-13 -1.9111e-11 -9.07838e-14 -4.83444e-13 -3.29891e-11 -1.59028e-13 -8.21378e-13
-1.91143e-11 -8.94531e-14 -4.70722e-13 -7.73091 5.47488 -0.417314 -1.91113e-11 -9.0447e-14 -4.80264e-13 -2.90554 0.965183
-0.102902 -1.91136e-11 -9.06458e-14 -4.79469e-13 -10.2786 0.731801 -0.0259123 -1.91076e-11 -8.94531e-14 -4.89805e-13
-2.22167 -0.733487 0.20892 -2.03217e-16 0.0 -3.88251e-19 0.199359 -2.70344 -0.0288713 -0.00656346 -0.653367 -0.0573627
-0.429977 3.35781 0.115501 2.03217e-16 0.0 3.88251e-19 -0.667256 -3.93835 0.0583947 0.00310878 1.58344 0.0241681 0.437817
2.35596 -0.0966915
-6.09799 82.9227 29.3751 -4.6261 -2.3821 9.5258 1.12113 -3.83681 -0.208598 -0.753063 -1.01916 -0.12772 -0.204573 1.72679
-1.94126 -0.0243454 2.67323 -3.8527 -0.0 0.0 0.0 -0.0 0.0 0.0 -2.32011e-14 -2.89445e-14 2.4817e-14 21.4215 1.46366
-2.74993 -2.78137 -1.06992 4.53206 0.107056 7.26942 0.992206 -3.59335e-10 8.49819e-11 3.22196e-10 -6.1418e-10 1.41398e-10
5.63565e-10 -3.59331e-10 8.49828e-11 3.22185e-10 7.73091 5.47488 0.417314 -3.59337e-10 8.49826e-11 3.22196e-10 2.90554
0.965183 0.102902 -3.59333e-10 8.49838e-11 3.22195e-10 10.2786 0.731801 0.0259123 -3.59342e-10 8.49854e-11 3.22199e-10
2.22167 -0.733487 -0.20892 -2.33758e-14 -2.8943e-14 2.45053e-14 -21.3092 3.58431 3.61727 4.72066 -2.25501 -7.24989
-0.0520236 5.06308 -0.701573 2.89021e-11 -1.90196e-10 -1.51561e-10 4.78409e-11 -3.31509e-10 -2.6295e-10 2.88997e-11
-1.90194e-10 -1.5155e-10 -7.73091 5.47488 -0.417314 2.89026e-11 -1.90195e-10 -1.51559e-10 -2.90554 0.965183 -0.102902
2.88993e-11 -1.90195e-10 -1.51558e-10 -10.2786 0.731801 -0.0259123 2.89053e-11 -1.90194e-10 -1.51568e-10 -2.22167
-0.733487 0.20892 -2.03217e-16 0.0 -3.88251e-19 0.964346 -6.01941 2.64839 0.00812651 4.38795 0.0333045 -0.722894 0.157148
-4.26721 2.03217e-16 0.0 3.88251e-19 1.57631 -8.01822 -3.57605 -0.00643354 7.81072 -0.102151 0.842749 -1.7792 3.24145
-5.96864 82.925 29.5392 -4.493 -2.1219 9.4487 2.60018 -7.88855 -0.420714 -1.46544 -2.08937 -0.246838 -0.283282 3.53992
-3.98989 0.328525 5.46824 -7.93512 -0.0 0.0 0.0 -0.0 0.0 0.0 -4.8847e-14 -5.994e-14 5.27525e-14 44.2209 2.99697 -5.66442
-5.53658 -2.17775 9.32542 -0.0623222 14.9582 2.09722 -7.16406e-10 1.72014e-10 6.81923e-10 -1.22371e-09 2.85808e-10
1.1923e-09 -7.16402e-10 1.72017e-10 6.81914e-10 7.73091 5.47488 0.417314 -7.16409e-10 1.72016e-10 6.81925e-10 2.90554
0.965183 0.102902 -7.16407e-10 1.72018e-10 6.81922e-10 10.2786 0.731801 0.0259123 -7.1641e-10 1.72018e-10 6.81928e-10
2.22167 -0.733487 -0.20892 -4.90017e-14 -5.99371e-14 5.24263e-14 -44.3264 7.33261 7.48046 9.09734 -4.51969 -14.9521
0.0298417 10.4121 -1.46156 8.81816e-11 -3.84265e-10 -2.99869e-10 1.48165e-10 -6.6983e-10 -5.20511e-10 8.81803e-11
-3.84262e-10 -2.9986e-10 -7.73091 5.47488 -0.417314 8.81829e-11 -3.84263e-10 -2.99868e-10 -2.90554 0.965183 -0.102902
8.81785e-11 -3.84262e-10 -2.99868e-10 -10.2786 0.731801 -0.0259123 8.81845e-11 -3.84263e-10 -2.99876e-10 -2.22167
-0.733487 0.20892 -2.03217e-16 0.0 -3.88251e-19 2.11574 -9.4929 5.52035 0.00616619 9.71808 0.130846 -1.56337 -3.18784
-8.91242 2.03217e-16 0.0 3.88251e-19 3.37617 -12.2585 -7.51992 0.0142647 14.3935 -0.241274 1.79585 -6.11351 6.80553
-6.01749 82.9214 29.3767 -4.5905 -2.4077 9.545 3.89196 -11.5017 -0.61923 -2.10244 -3.0441 -0.354358 -0.262143 5.15654
-5.84488 1.00603 7.87416 -11.6569 -0.0 0.0 0.0 -0.0 0.0 0.0 -6.59229e-14 -8.68108e-14 7.42523e-14 64.6853 4.35048
-8.27108 -7.84097 -3.12955 13.6145 -0.483136 21.781 3.17992 -1.0398e-09 2.55141e-10 9.67549e-10 -1.77654e-09 4.24543e-10
1.69237e-09 -1.0398e-09 2.55145e-10 9.67542e-10 7.73091 5.47488 0.417314 -1.03981e-09 2.55144e-10 9.67554e-10 2.90554
0.965183 0.102902 -1.03981e-09 2.55147e-10 9.67549e-10 10.2786 0.731801 0.0259123 -1.0398e-09 2.55145e-10 9.67556e-10
2.22167 -0.733487 -0.20892 -6.60551e-14 -8.6807e-14 7.39133e-14 -65.2961 10.5802 11.0202 12.4869 -6.33272 -21.873
0.227258 15.18 -2.17234 1.24928e-10 -5.70406e-10 -4.53717e-10 2.09501e-10 -9.94209e-10 -7.87207e-10 1.24928e-10
-5.70402e-10 -4.53709e-10 -7.73091 5.47488 -0.417314 1.2493e-10 -5.70403e-10 -4.53717e-10 -2.90554 0.965183 -0.102902
1.24925e-10 -5.70403e-10 -4.53716e-10 -10.2786 0.731801 -0.0259123 1.24931e-10 -5.70405e-10 -4.53724e-10 -2.22167
-0.733487 0.20892 -2.03217e-16 0.0 -3.88251e-19 3.44084 -12.5464 8.15502 -0.0114717 14.4705 0.221361 -2.77051 -6.08147
-13.1073 2.03217e-16 0.0 3.88251e-19 4.46605 -15.9324 -11.1741 0.061438 20.2649 -0.375151 3.10729 -9.9064 10.0645
```

Figure 1.9: The second half of a BVH file.

Representation of Human Motion Data

The motion capture data over the character's skeleton can be represented as the position and orientation of the character's root in conjunction with the orientation of each joint. More specifically, considering that P and Q is the position and the orientation of the character's root, and r_i for $i = 1, ..., N$ the orientation of the $i - th$ joint of the character, a posture, which is the positioning of the character, can be represented as:

$$posture = \{P, Q, r_1, ..., r_n\} \tag{1.1}$$

Although, since a posture describes a static pose, as well as since it is required the ability of representing the human posture, as it evolves during the time, a motion file is represented as the position and orientation of the character's root in conjunction with the orientation of each joint at each time step. Therefore, a motion sequence over the character's skeleton for $t = 1, ..., T$ is represented as:

$$motion = \{P(t), Q(t), r_1(t), ..., r_n(t)\} \tag{1.2}$$

Based on this representation it is now possible to handle the data files, such as being able to synthesise new motion sequences or to edit the motion sequences, by retrieving the necessary information.

Issues with Motion Capture Data

Even if the MoCap technologies provide the ability of capturing the desired motion sequences, three major disadvantages may characterise the MoCap process. Firstly, those devices can be characterised for its high cost. This cost is not always a problem especially for big companies. Although, for homemade studios and small companies is impossible to have a high quality MoCap system. Secondly, even if a high-cost MoCap device captures highly realistic human motions, there are undesired effects that are produced during the MoCap process. Typical examples of this are the undesired footsliding effect Kovar et al. (2002b), and the noise produced by the sensors Lou and Chai (2010) during the capturing process.

More specifically, with the footsliding effect it is meant the inability of the foot to stay static in the ground. Hence, computational solutions for removing this undesired effect, such as Kovar et al. (2002b) Lyard and Magnenat-Thalmann (2007), provid the ability of removing this effect while the motion data retains the desired naturalness of human motion. Similarly, there are computational methods Lou and Chai (2010) for removing the undesired effect of noise that is capturing simultaneously with the motion data. Thirdly, since the captured motion sequences should be placed in a virtual environment, there are issues related with the ability of fulfilling exactly the required tasks (see Kallmann et al. (2003) Wang and Verriest (1998)). For example, considering a simple motion where a real human performs a task for reaching an object in the real space, the virtual character may not by able to reach the exact position of the object due to the inability of the motion database to provide all possible reaching points where the character can perform within the virtual environment. This simple problem can be efficiently solved by using motion interpolation techniques such as Wiley and Hahn (1997a) Mukai and Kuriyama (2005) or motion blending techniques such as Kovar and Gleicher (2004) Huang and Kallmann (2010a). In cases that is required more competitive human motion parameterisation process, solutions for editing Raunhardt and Boulic (2009) Carvalho et al. (2013) Gleicher (2001) and synthesising Shin (2006) Carvalho et al. (2007) the exact desired motion have been examined as well during the past years.

1.3 Inversing the Content Creation for Animation

Animating virtual characters is a quite complex process. This is since, virtual characters as humans consists of numerous bones and joints. Therefore, animating virtual characters by using traditional keyframe techniques requires firstly the knowledge of special software packages as well as requiring experience and talent. A simple representation of the content creation pipeline for animating virtual characters is illustrated in Figure 1.10.

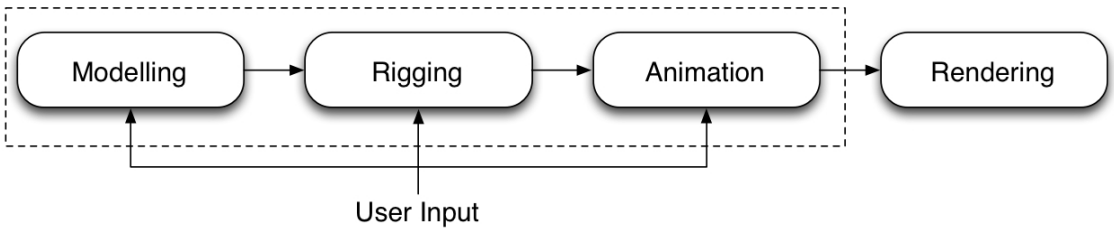

Figure 1.10: The standard content creation pipeline for animating virtual characters.

In general the animation content creation pipeline consists of the followings. Firstly, it is required an experienced designer in order to model or to sculpt the morphology of the character. Then, it is required an expert rigger that will rig properly the virtual characters. The third step of the content creation pipeline is based on the ability of animating the virtual character. Thus, another experienced user is required to create as realistic as possible movements for the character. Finally, having created the content, the character can be placed in the desired scene, and the whole scene can be rendered.

Separate part of the animation content creation pipeline requires experienced developers. However, techniques for solving each of the separate parts of the animation pipeline have been developed during the past years by the research community. Rather than modelling the shape of the character by hand, it is possible to simply capture and reconstruct its shape using a 3D scanner (surface reconstruction) Li et al. (2009) Allen et al. (2003). Additionally, rather that rigging the character by hand, it is possible to automatically rig Baran and Popović (2007) Pan et al. (2009) the character based on example-based rigging techniques. Finally, rather than creating the desired movements of the virtual characters, it is possible to directly record the animation Deutscher et al. (2000) Deutscher and Reid (2005) using a MoCap system (motion capture). Then, those motion sequences can then easily be retargeted automatically Bindiganavale and Badler (1998) Monzani et al. (2000) to any character, placed in a 3D environment and rendered within the desired virtual scene.

Based on those approaches for creating the required content, it can be stated that an

alternative way to customizing animation controls by hand is to train the system with inputs of real artists so that it is capable of performing the same task automatically in the future (training). Hence, all those methodologies that use the inverse process of generating the computer animation can be termed animation reconstruction Li (2010). Figure 1.11 illustrates the inverse content creation pipeline for animating virtual characters.

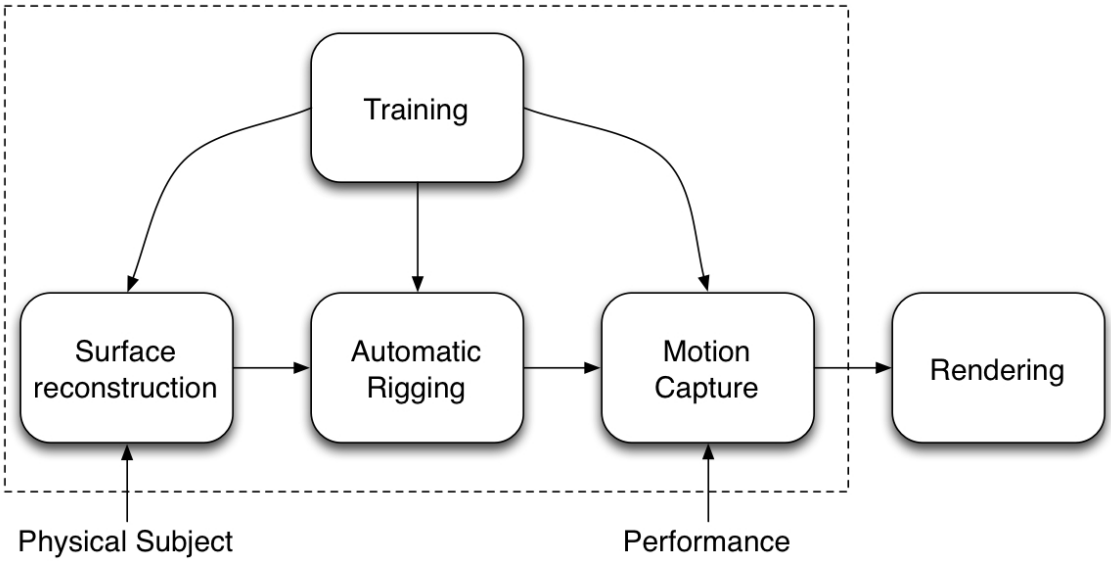

Figure 1.11: The inverse content creation pipeline for animating virtual characters.

1.4 The Presented Research

Methodologies for synthesising human motion data provide the ability of animating virtual characters quite easily and fast, although the final result might not be realistic enough. Thus, the presented research examines the ability of synthesising motion sequences while the realistic representation of the synthesised motion is increased. Based on the ability of efficiently and easily handling the necessary parameters, three different techniques for animating virtual characters were developed. Specifically, it is presented:

- the ability of synthesising the character's locomotion based on easily specified footprints,
- a methodology for synthesising style variations of the full-body motion of a character during the performance capture process,
- a methodology for the virtual character's finger motion synthesis.

The first data-driven method that is examined in this thesis is the ability of synthesising the character's locomotion. More specifically, in cases that it is desired the ability of

synthesising realistic representation, the so-called human-like, of the virtual character's motion planning, the following two questions raised:

- **Q1:** Is a reasonable way to control motion synthesis from a motion database?
- **Q2:** Which is the best way of synthesising motion that includes changes in movement type?

The advantage of the presented locomotion composition approach consists of two parts. Firstly, the system is able to synthesise a valid transition process while the character follows the footprints that are located in the 3D space. The second advantage of the method is based upon the ability of the system to automatically recognising the action that should be generated. For achieving this automatic action recognition, firstly by introducing the footprint patterns, the patterns that are responsible for separating each locomotion behaviour, as well as by using the parameters retrieved from the action transition graphs, which is the number of steps where the character requires for traversing to a specified task, the system is responsible for automatically synthesising the desired locomotion of the character.

However, virtual characters would be able to animate according to different style variations, instead of performing specified behaviours and styles. Thus, in the second part of the presented research it is examined the ability of synthesising in real-time style variation of the input motion of a performer. Capturing the desired behaviour of the performer based on the role as well as on the target character's capabilities, requires experienced actors. In cases where a film company is responsible for producing a movie, actors are employed for performing the desired motions. In cases where home-made production requires the ability of capturing stylistic representations of motion sequence, since the cost of hiring actors in not always affordable, techniques that are responsible for synthesising in real-time the desired stylistic variation of a motion should be investigated. Even if various techniques provide the ability of motion style transfer, the key disadvantage that characterise those techniques is that those methods work off-line. Therefore, based on the requirement of synthesising in real-time the desired style variation of an input motion during the performance capture process, the following questions raised:

- **Q3:** Is it possible to convert the off-line motion style transfer process in real-time during the performance capture?
- **Q4:** How possible is to parameterise the input motion with the desired stylistic component?

Therefore, based on these questions a methodology that is responsible for synthesising in real-time a full-body motion sequence with predefined stylistic variations is presented. By using statistical motion analysis and synthesis techniques, and a parameterised maximum a posteriori (MAP) framework it is possible to convert the input postures of the performer to stylistic postures that are mapped directly to a virtual character.

In this case, it can be stated that the presented methodology extends previous research on motion style transfer and performance-driven full-body character control such as Brand and Hertzmann (2000) Hsu et al. (2005) Chai and Hodgins (2005) Liu et al. (2011b). Although, comparing with the previous research, rather than retargetting the stylistic content in off-line processes, the proposed solution investigates the ability of synthesising the desired stylistic variation during the motion capture process. This is achieved by using a four-step motion synthesis process: (i) capturing individual stylistic behaviours, (ii) aligning and synchronising the captured motion sequence, (iii) building the correlation between two individual motion styles, and (iv) generating a parameterised likelihood function over the statistical motion model for synthesising in real-time the desired motion sequence.

With both the aforementioned techniques the synthesis of a virtual character in a natural looking way can be improved. However, the required realism of motion sequences can be further improved by assigning details to the virtual character's body, such as the motion of the character's fingers (which is often missing from the motion capture database). Therefore, the final technique that is examined is based on the ability of estimating and synthesising the motion of a character's fingers. In general, the presented methodology is coming to make animators life easier, as well as to reduce the time for designing or capturing the desired motions. Specifically, considering that always it is desired a realistic representation of human motion and perceptual studies (see references Hoyet et al. (2012) Jörg et al. (2010) Jörg et al. (2012b) Samadani et al. (2011a)) have shown that the addition of the motion of the fingers enhances the realism of the animated sequence. Moreover, the production of highly realistic characters and movement can be beneficial in various applications of virtual reality such as in virtual telepresence. Thus, in this case it was firstly examined an efficient methodology for estimating the motion of the fingers while the computations are minimised, as well as the ability of synthesising the motion of the character's fingers in real-time, during the performance capture process while low-cost motion capture devices are used. Therefore, the research questions that arise are the followings:

- **Q5:** Is it possible to synthesise realistic finger movement when this information is not contained in the motion database?

- **Q6:** Is it possible to minimise the computation cost that is required for estimating and synthesising the motion of a character's fingers?

- **Q7:** How possible is the estimate in real-time the motion of the character's fingers, during the performance capture process?

Although, as the increase of motion data provide the ability of using existing motion sequences for animating virtual characters, it is presented a methodology of estimating the character's motion based on parameters where the character's wrist can provide. In general, such a methodology can be quite beneficial in cases where meaningful motion sequences over the character's hand are required. The presented research extends the current trend that is the use of spatial information for estimating the character's finger motion. This is achieved by introducing the ability of estimating the motion of the fingers based on motion features. Therefore, based on a cross validation study, it is shown that such a methodology shows significant benefits. Finally, it should be mentioned that this methodology can be quite beneficial in cases that is required a fast as well as an efficient way for estimating the character's finger motion in real-time. Thus, in addition to the off-line motion synthesis process, a simple extension of this method is presented that estimates and synthesise the motion of a character's fingers in real-time.

1.5 Contribution

Summarising, the contribution of this research thesis is presented below. Specifically, this thesis contributes in the following four parts:

- A character animation methodology based on existing motion data that contains sequences which are responsible for synthesising the locomotion of the virtual character based on predefined footprints. This methodology does not only synthesise the required motion of a character based on predefined footprints, but also includes a novel technique for determining the behaviour and transitions of the animated character between different motion types.

- A human motion synthesis process for animating virtual characters with style variations is introduced. The proposed method takes the benefit of building a correlation

between existing motion styles. Based on those styles the system is responsible for animating the virtual character while it keeps the desired stylistic content.

- The ability of estimating the character's finger motion based on parameters that the character wrist provides. Therefore, a hybrid estimation method that is based on motion features is responsible for synthesising meaningful motions of the character's fingers.

- Since the finger motion estimation process based on motion features provides the ability of faster computations, a simple extension of the aforementioned methodology is presented that estimates and synthesise in real-time the required motion.

1.5.1 List of Publications

During the doctoral studies, the following papers produced by the author of this thesis. The listed papers published/submitted in journals and conferences related to computer animation, computer graphs, and virtual reality.

Journal Papers

1. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Finger Motion Estimation and Synthesis for Gesturing Characters" Submitted in Graphical Models Journal, Elsevier, 2014.

2. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Performance-Driven Hybrid Full-Body Character Control for Navigation and Interaction in Virtual Environments" Submitted in Transactions on Visualization and Computer Graphics (Proc. of IEEE VR 2015), IEEE Press, 2014.

3. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Performance Animation with Style Variations" Submitted in Journal of Multimedia, Academy Publisher, 2014.

4. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Footprint-Driven Locomotion Composition" International Journal of Computer Graphics & Animation, AIRCCSE Publishing Corporation, 2014 (to appear).

5. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "The Minimum Energy Expenditure Shortest Path Method" Journal of Graphics Tools, Taylor & Francis, Volume 17, Issue 1-2 pp. 31-44, 2013.

6. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Splicing of Concurrent Upper-Body Motion Spaces with Locomotion" Procedia Computer Science Journal, Elsevier, Volume 25, pp. 348-359, 2013.

Conference Papers

1. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Analysing and Segmenting Finger Gestures in Meaningful Phases" To Appear in IEEE International Conference on Computer Graphics, Imaging, and Visualization, 2014.

2. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Efficient Hand-Over Motion Reconstruction" In Proc. of 22nd International Conference on Computer Graphics, Visualization and Computer Vision, pp. 111-120, 2014.

3. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Data-Driven Motion Reconstruction Using Local Regression Models" In Proc. of 10th International Conference on Artificial Intelligence Applications and Innovations, pp. 364-374, 2014.

4. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Evaluating the Covariance Matrix Constraints for Data-Driven Motion Reconstruction" In Proc. of ACM/Eurographics Spring Conference on Computer Graphics, pp. 99-106, 2014.

5. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Rethinking Shortest Path: An Energy Expenditure Approach" In Eurographics Proc. of Virtual Reality Interaction and Physical Simulation, pp. 35-39, 2013.

6. C. Mousas, P. Newbury, and C.-N. Anagnostopoulos "Measuring the Steps: Generating Action Transitions Between Locomotion Behaviours" In IEEE Proc. of Computer Games, pp. 31-35, 2013.

7. C. Mousas, and P. Newbury "Real-Time Motion Synthesis for Multiple Goal-Directed Tasks Using Motion Layers" In Eurographics Proc. of Virtual Reality Interaction and Physical Simulation, pp. 79-85, 2012.

1.6 Thesis Structure

The rest of this thesis is divided in the following chapters. In Chapter 2 the literature review on computer animation techniques that are related to this thesis is presented. In Chapter 3 a methodology for the character's locomotion composition process is presented.

A methodology for synthesising in real-time, during the performance capture process, motion sequences that keep the desired style variations is presented in Chapter 4. The ability of estimating the character's finger motion based on parameters extracted by the character's hand is presented in Chapter 5. Finally, conclusions, and the potential future work are discussed in Chapter 6.

Chapter 2

Literature Review

In this chapter, the literature review related to human motion synthesis for animation virtual character is presented. More specifically, firstly, it is presented a general overview of common methods for the animation of virtual characters. Then, the rest of this chapter is focused on methods for each aspect covered in this thesis. It is presented previous work that covers the statistical motion analysis and synthesis, the motion planning techniques, the performance animation techniques, the style motion synthesis, and the finger motion synthesis. In general motion synthesis for animating virtual characters can be divided in two major categories: the non-computational, and the computational methods. Especially for the computational methods, it is possible to split the different methodologies again in lower-level categories Glardon (2006) that are the kinematics, the physics, and the data-driven methods. Each one of these methods has its own subcategories, although an extended focus is given to the data-driven methods since these methods are more related to this thesis. Hence, the following subsections present previous work on human motion synthesis based on these categories.

2.1 Non-Computational Methods for Animating Virtual Characters

The most common methodologies that are used for animating virtual characters, is based upon the ability of using specific software solutions. These solutions have been developed for providing tools and functionalities for animating virtual characters. Generally, the methods that are used for animating virtual character are generally called “hand-driven animation”, which is the oldest and simplest way for generating the desired animation Sturman (1985). Thus, by using animation software solutions, the animator designs manually

the postures of the articulated character by defining the positions and the orientations of the character's joints at specified time steps. This method for animating virtual characters is called "key-framing", since the animators specifies each posture of the character at different key-frames. In this case, the software is responsible for generating a smooth interpolation between the corresponding key-frames that have been placed by the user. However, the key-framing approach for animating virtual characters is always depended on the technical and artistic skills of the animator. Hence, reproducing highly realistic animation for virtual characters can be quite difficult. This is since, every degree of freedom (DOF) of character's skeleton should be naturally animated, composing a highly realistic and natural looking motion.

Therefore, software solutions such as Autodesk's Maya Autodesk (2014a), and Motion Builder Autodesk (2014b), Reallusion's iClone Reallusion (2014), Blender Blender (2014), and Poser SmithMicro (2014), and many more, are available for helping the animators by providing tools for the animation pre-visualisation, interpolation methods, rigging methods etc. for manipulating and animating the virtual characters easier. A simple example of Autodesk's Maya Autodesk (2014a) developing environment is shown in Figure 2.1. Although, even if these solutions provide the ability of designing the desired animations, it is characterised as a time consuming process that is required for animating the virtual characters. Thus, as the time that is required for producing animations for complex articulated characters, as well as while the duration of the required animation increases, the key-framing method is not recommended.

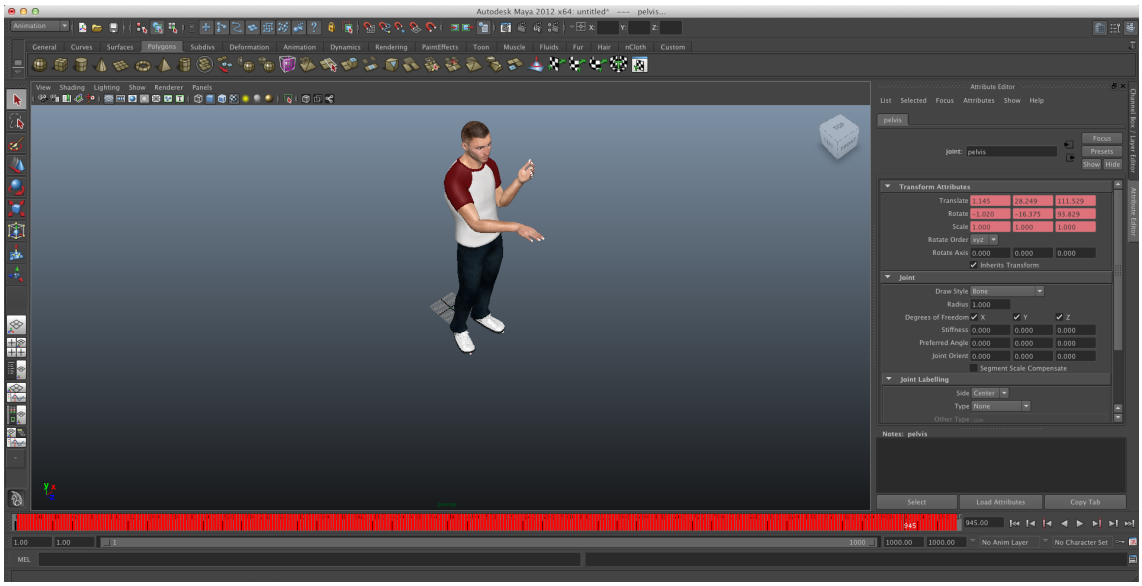

Figure 2.1: The developing environment of Autodek's Maya Autodesk (2014a).

2.2 Computational Methods for Animating Virtual Characters

Due to the previously mentioned drawbacks of hand-driven animation the ability to automatically animate virtual characters by defining the appropriate rules has attracted the research community. Therefore, various methods have been developed providing the ability of efficiently animating the virtual characters. More specifically, those methods can be divided in three major categories, as proposed by Glardon Glardon (2006), Van Welbergen et al. Van Welbergen et al. (2010), and Multon et al. Multon et al. (1999), the kinematics, the physics, and the data-driven methods that includes signal processing, motion warping, motion interpolation etc. Following, a short explanation of kinematics, and physics methods, as well as an extended explanation of the data-driven methods, since these are more related to the presented research, are presented.

2.2.1 Kinematics Methods

The kinematics methods for animating virtual characters rely on the biomechanical knowledge, and combine direct and inverse kinematics (IK) knowledge Multon et al. (1999). One of the most important drawbacks of these methods is the inability to provide realistic motions. This is since, kinematics do not necessarily preserve physical laws. For example, even if it is possible to position easily each of the character's joints in the 3D space, the posture of the character may not be realistic enough. Moreover, since the human body consists of various limitations, kinematics-based animation does not ensure that the motion of the character lies within these limitation. Hence, unrealistic or unnatural postures can be synthesised. Additionally, the generated motion that results from kinematics solutions may have a robotic-like representation. Although, the kinematics is one of the most important methods for motion synthesis and has been studied extensively, especially in robotics Baker and Wampler (1988) since they are originated from this field. In general, there are two main approaches for computing the kinematics: the analytical and the numerical methods.

The analytical solvers are based on the ability of projecting the vector loop equation of a mechanism onto the axes of a non-moving Cartesian frame. This projection transforms a vector equation into two algebraic equations. Then, for a given value of the position (or orientation) of the input link, the algebraic equations are solved for the position and/or orientation of the remaining links. On the other hand, numerical solvers use the Jacobian

matrix to find a linear approximation to the IK problem. The Jacobian solutions linearly model the end effectors' movements relative to instantaneous system changes in link translation and joint angle. For a detailed explanation of kinematics solvers the reader is referred to Aristidou and Lasenby (2009). In general, the analytical methods Chung and Hahn (1999a) Tolani et al. (2000) are very efficient, but the main limitation is that they cannot solve the complicated structures such as an articulated figure. The numerical methods Komura et al. (2003) Wampler (1986) can overcome the limitations of the analytical solution of kinematics and are suitable to any model with fast implementation, although the required human-like realism is not provided.

Since the kinematics is among the first established methods for animating virtual characters, various methods have been developed that generate motion patterns for virtual characters. Hence, the solutions provided by Zeltzer Zeltzer (1982) Zeltzer (1983) are able to control the synthesised human gait. This is achieved by defining the control parameters over a state machine that represent a bundle of key-frames, which are then interpolated to produce the desired walking motion. Boulic et al. Boulic et al. (1990) developed a walking engine that allows animating virtual character driven by two high-level parameters: the linear and angular velocity of the limbs. Finally, another solution that offers a higher-level of control has been proposed by Sun and Metaxas Sun and Metaxas (2001). This method adapts the walk to uneven terrains by deforming the original motion that is generated in a 2D space. This is achieved by controlling the step length and step height parameters. Although, the basic limitation of motion deformation is the inability of editing in a natural-looking way the motion of a character, since the deformations does not ensures the physical validity of the resulted motion.

However, even if kinematics-related methods cannot produce highly realistic results, as mentioned previously, there are issues in computer animation that can be efficiently solved by using kinematics. Hence, two very interesting solutions that kinematics methods provide is that ensures that the feet do not penetrate into the ground, as well as the ability of removing the foot sliding effect, since it is possible to constraint the feet end-effector to stay in specific positions during the motion synthesis process. More specifically, Bruderlin and Calvert Bruderlin and Calvert (1989) proposed a method that changes the character's origin so that its origin is always fixed on the supporting foot during the walking motion. Although, since the leg in their system contains only one joint, the resulted animations may not be natural looking enough. Hence, for improving this undesired motion, they added more DOFs to the character's leg model, resulting to produce smooth and parameterised

walking gaits Bruderlin and Calvert (1993). A similar methodology has been applied to running motions Bruderlin and Calvert (1996). Additionally, Chung and Hahn Chung and Hahn (1999a) proposed a method where the foot position of the supporting leg is controlled prior to the swing phase of the leg (see Figure 2.2). The foot sliding removal methods, which are methods for removing the inability of the foot to stay static on the ground, proposed during the past years. Hence, computational solutions that uses kinematics solutions for removing this undesired effect, such as the one proposed by Kovar et al. Kovar et al. (2002b), provides the ability of removing this effect while the motion data retains the desired naturalness of human motion, by enforcing kinematics solution.

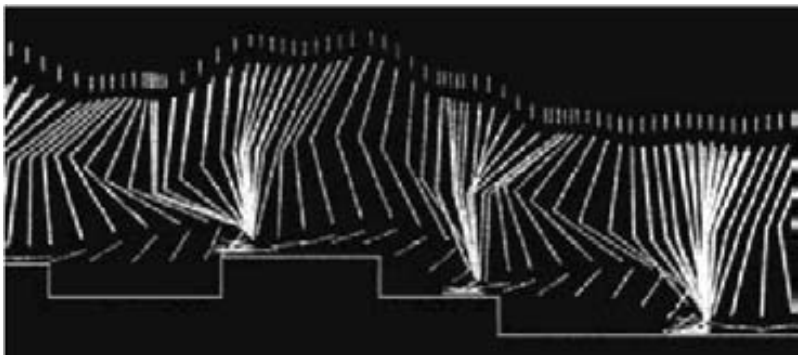

Figure 2.2: A kinematics-based approach for synthesising walking motion as it is proposed by Chung and Hahn Chung and Hahn (1999a).

2.2.2 Physics Methods

Physics-based methods can be described as more generic that adapt better to the environment. This is since, with physics-based methodology the synthesised motion is able to follow various laws that characterise the human motion. Moreover, physical interactions between the character and the environment can be synthesised easily. A typical example is the ability of a character to walk in a unnerving terrain. Hence, physics laws can be able to understand the required human skills and to synthesise a physically valid motion.

In general, those methods keep the advantage of simulating the real human movements, which are governed by the laws of physics. This is achieved by simulating the actual physical forces/torques that act on the articulated body. Those methods compute each body part displacement as a function of their mass distribution and of the torques generated at each joint.

In general, most physics-based animation methods deal with automatic generation of animation with minimal user inputs. Space-time optimization approaches automatically create physically plausible animation by solving optimization problems subject to phys-

ical and other constraints Liu et al. (2005). Researchers have constructed such dynamic controllers to let characters perform swimming Yang et al. (2004), stable walking cycles Laszlo et al. (1996), reactive motions such as falling Faloutsos et al. (2001a), and other motions such as breathing and grasping Zordan et al. (2004). Such approaches can work in conjunction with kinematic animation Shapiro et al. (2003) and motion capture Zordan et al. (2005). Finally, researchers have also used physical simulation to create realistic secondary motions Neff and Fiume (2006), such as the motion of the character's hair Derouet-Jourdan et al. (2013) and clothes Kim et al. (2013).

During the past years, various hybrid methods Girard and Maciejewski (1985) that combine kinematics and physics for animating virtual characters had been proposed. Specifically, Hodgins and Wooten Hodgins and Wooten (1998) implemented simple athletic maneuvers based on Proportional Derivative (PD) servos that compute the joint forces and torques. The aforementioned method was improved by Wooten et al. Wooten and Hodgins (1996) by allowing the switch from one to another controlled by generating transitions between jump and landing motions, while the character keeps its balance. Faloutsos et al. Faloutsos et al. (2001b) proposed a methodology where the balance is also controlled such as being able to generate recovering motion from a fall (see Figure 2.3).

Figure 2.3: A physical based approach for recovering from a fall as it is proposed by Faloutsos et al. Faloutsos et al. (2001b).

The key disadvantage is its computation cost that is required, since a large number of equations must be solved, limiting the number of characters that can be simulated at the same time. Another difficulty of this approach consists on the ability of specifying all the required forces that should be applied to a system in order to generate the desired movement. Although, in the biomechanics literature even if it is possible to be found the average of mass distribution, the determination of joint torques for achieving a particular motion is quite difficult to model Winter (1990). It should be noted that physics simulations are always depended on human activities that can be described by physical laws. Although, since human motion is non-linear it is impossible to be described efficiently each motion that can be performed by real humans. For achieving highly realistic animated characters highly complex mathematical explanation of human motion is required. It is

not always possible to simulate a large number of characters in real-time, making the physic-based methodologies quite computational expensive for real-time applications. As it is always desired the ability of synthesising as natural looking motions as possible more complex composite controllers have to be designed, which will be able to synthesise a full range of human like motions.

2.2.3 Data-Driven Methods

In general, the data-driven methods use input data that is provided by the motion capture systems in order to produce the desired animation. These methods can be characterised as the most natural looking, since the desired animation results from real humans performing the desired actions. The data-driven methods vary, as the requirements of fulfilling a task for the virtual character vary. This is since, different methodologies are responsible for editing existing motion data such as fulfilling specified constraints, or synthesising the motion data such as enabling the character to follow a large number of different behaviours. Hence, methodologies that examine the ability of blending, interpolating, warping etc. have been proposed during the past years Wang et al. (2006). A detailed explanation of the most interesting data-driven methodologies for editing or synthesising the human motion data, are presented below.

Signal Processing

Signal processing methods try to represent the motion in the non-Euclidian space, such as being able to benefit from other properties that may characterise a motion sequence. This means that instead of editing the motion sequences based on the position and orientation of each character's joint, it is possible to be edited by handling the representation of the motion data by its features. Hence, Bruderlin and Williams Bruderlin and Williams (1995) by applying widely used methods that are related with image and signal processing, such as multi-resolution filtering, multi-target interpolation, waveshaping and displacement mapping, tried to design, modify, and adapt the motion sequences, such a being able to produce new motion sequences to fulfil the user specified constraints. Unuma et al. Unuma et al. (1995) represented the motion sequences based on Fourier series. In this case, the motion sequences can be compared and qualified the characteristics between similar motions such as the tiredness, sadness, and joy. Amaya et al. Amaya et al. (1996) expressed the differences between an emotional and a neutral motion as an emotional transform. This latter is computed as a warp that can be applied to the timing and amplitudes of the

neutral motion. These warps can be efficiently applied to other neutral motion types in order to add similar emotional content. An example over this is the approach proposed by Perlin Perlin (1995). In this methodology by using principles from procedural texture synthesis, subtle human movements such as shifting of weight and fidgeting while standing were created.

The basic advantage of this methodology is based upon the ability of editing the motion data such as being able to fulfil various constraints. However, the key disadvantage is based upon the ability of editing specific properties of the motion. Although, since the editing process by using signal processing techniques is implemented in motion data, it is ensured that the new motion lies within the natural looking space.

Motion Warping

This approach was introduced by Witkin and Popović Witkin and Popovic (1995). In this method, the animator interactively defines a set of key-frames inducing a set of constraints, that are used to derive a smooth deformation preserving the fine structure of the original motion. However, motion warping methods are purely geometric methods and operate on each DOF independently, without understanding the motion structure. They are not well suited for adjustments requiring coordinate movements, such as grasping actions with modification of object location. In such cases, not only the joint hand is affected, but also other joints like the elbow or shoulder, which results on producing unrealistic motions.

Motion Blending

Motion blending is created by pre-processing "similar" example motion primitives, typically such that they correspond in time (especially at key events such as foot plants) and space (e.g. root rotation and position). The process of time-aligning corresponding phases in motion primitives, is called time warping Bruderlin and Williams (1995). Kovar and Gleicher Kovar and Gleicher (2003) present an integrated method called registration curves to automatically determine the time, space and constraint correspondences between a set of motion primitives and provide an overview of previous methods used for this. The blending weights are not necessarily computed in the Euclidean space. Dimensionality reduction techniques such as the principal component analysis provides the ability of computing the blend weights the motion sequences in the latent space Carvalho et al. (2013). The advantage of such a methodology is based upon the ability of synthesising new motion sequences while the specified constraints are fulfilled, while the required naturalness of the

synthesised motion is kept.

Motion Interpolation

In cases where character should follow a path from a start to a goal position, it is not always possible to capture the locomotion behaviour of the character, since there are limitations about the dimensions of the real space where the motion capture process evolves. The motion interpolation methods provide the ability of synthesising a smooth motion sequence where different whole body motions are placed one after other in the row. In general motion interpolation methods Park et al. (2004) Rose et al. (1998) are often used to produce smooth transitions between two different behaviors, for example transitions from walking to running. However, it is often chosen empirically which motions to blend, how to align the motions in time, how long the blending window should be and how the blending weights should be set in order to produce natural transitions. In general, the blending method Wiley and Hahn (1997b) creates a motion primitive by interpolating a family of example similar motion primitives. The animation parameters are interpolation weights and a selection of the example motion primitives to interpolate. The interpolation does not need to take place in the Euler space, but can also be done in, for example, the principal component Lim and Thalmann (2002) or in the Fourier Unuma et al. (1995) domain. In general, one can only interpolate between poses that "resemble" each other. When this is not the case, visual artefacts such as foot sliding may appear. Among the most well known approaches in the motion graphs, proposed by Kovar et al. Kovar et al. (2002a). Finally, other powerful methodologies for motion interpolation are proposed by Rose et al. Rose et al. (1998), Wiley and Hahn Wiley and Hahn (1997b), and Mukai et al. Mukai and Kuriyama (2005). An example of a motion interpolation method is illustrated in Figure 2.4.

Motion Transplantation

Current motion capture technologies cannot provide the direct capture of the whole human body, which includes the motion of the fingers and the motion of the face. Hence, methods that separately captures each of those motions, and assembly them together have been attractive to the research community during the past years. Those methods generally called "motion transplantation" Ikemoto and Forsyth (2004) Van Basten and Egges (2012) Ha and Han (2008) are responsible for combining the desired motion sequences on character's different body parts, synthesising a high quality and realistic motion sequence (see Figure

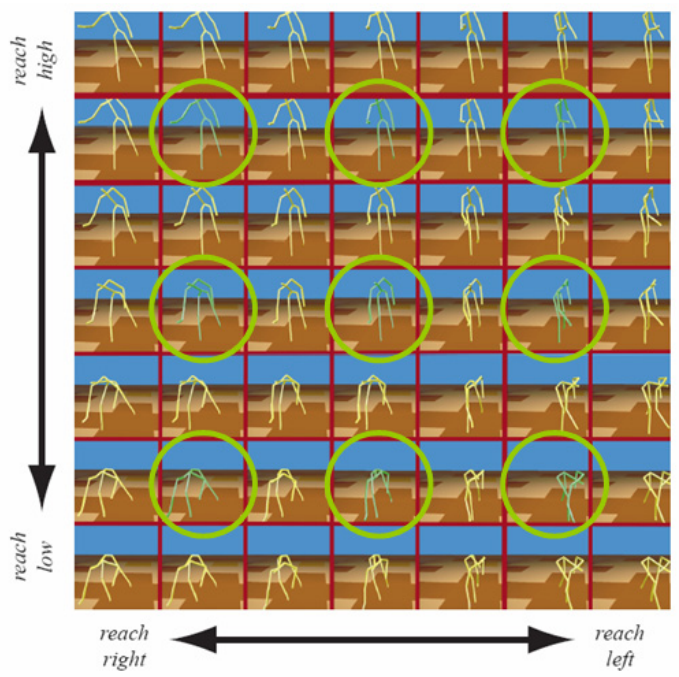

Figure 2.4: Motion interpolation between variations of similar motions as proposed by Rose et al. Rose et al. (1998). A reach sampled across two axes of the goal position for the hand. The green figures are the example motions. The rest are created through the Verb/Adverb mechanism.

2.5). Although, since a simple combination of different motion sequences that correspond at different body part may not generate the exact natural looking motion, methods for synchronising the separate motions based on timewarping methodologies Kovar and Gleicher (2003) Hsu et al. (2005), as well as methods for aligning Horn (1987) Schwartz and Rozumalski (2005) in a correct manner the different partial motion sequence, have been examined as well. Finally, it should be mentioned that the motion transplantation method combined with the motion graphs Kovar et al. (2002a) synthesis process by Ng et al. Ng et al. (2010). Hence, it was possible to use partial motion sequences within a long stream of motions, enhancing the resulted behaviours that can be synthesised by the character.

2.2.4 Evaluating Motion Synthesis Methods

Each of the aforementioned methodologies has its advantages and disadvantages. A direct comparison is quite complex since, different aspects influence the decision of choosing the most appropriate method for animating a virtual character. Although, trying to figure out which of the methods is more appropriate for specified reasons the following three figures (Figure 2.6 to 2.8) illustrates a simple comparison between the control freedom against the motion realism, the computational time against the motion realism, and the developing

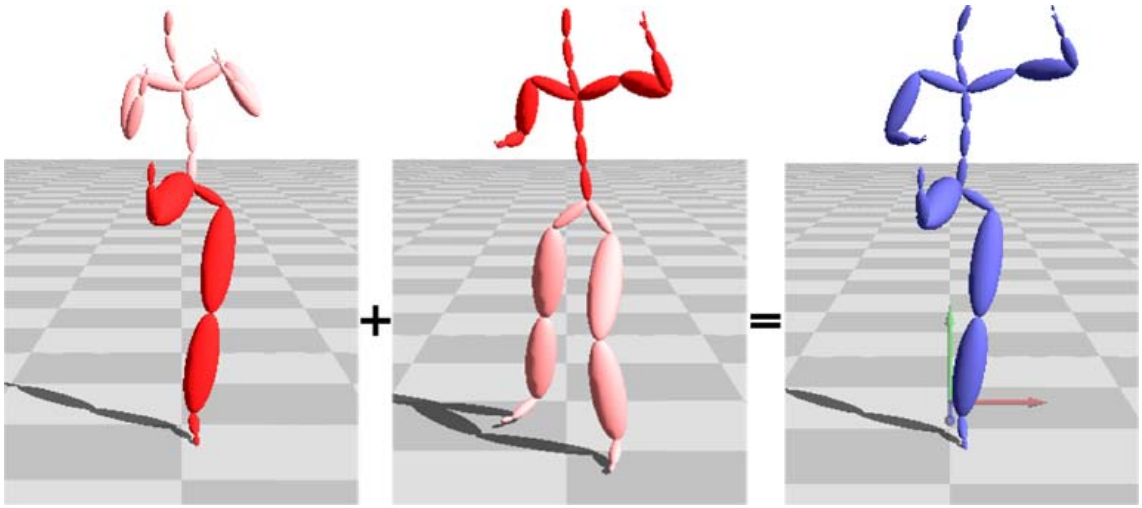

Figure 2.5: Transplanting upper-body actions with lower body action, as it is proposed by Ha and Han Ha and Han (2008). Specifically, by combining the lower-body motion (left) with the upper body motion (middle), the system synthesise a new motion sequence (right) that keeps the required components of the aforementioned two motions.

time against the motion realism, as resulted from Glardon (2006) Van Welbergen et al. (2010). It should be noted that with "control freedom" it is meant the ability of the user to manipulate each body part of the character such as being able to synthesise the required posture of the character.

Based on this simple comparison, the following can be stated. While it is required highly realistic animated sequences the most appropriate methods for animating virtual characters are the data-driven, and the hand-driven methods. Although, depending on the computational time, as well as on the required control over the character, it is possible to distinguish those two methods. The kinematics and physics methods lag on the control freedom, on computational time, and on motion realism. Hence, even if kinematics methods can be quite beneficial in specific domains, in cases that is desired natural looking motions that run on interactive frame rates (such as in video games) those methods could not be the primary choice for animating virtual characters. Therefore, it is possible to conclude that the data-driven methodologies are the best way for animating a virtual character where the required realism of a new synthesised motion sequences is high, as well as the computational cost kept to a minimum compared with the other solutions.

2.3 Statistical Analysis and Synthesis of Human Motion

In this section it is presented methods that uses models that are learned from the statistical variation of example motion primitives. In general those methods uses artificial intelligence

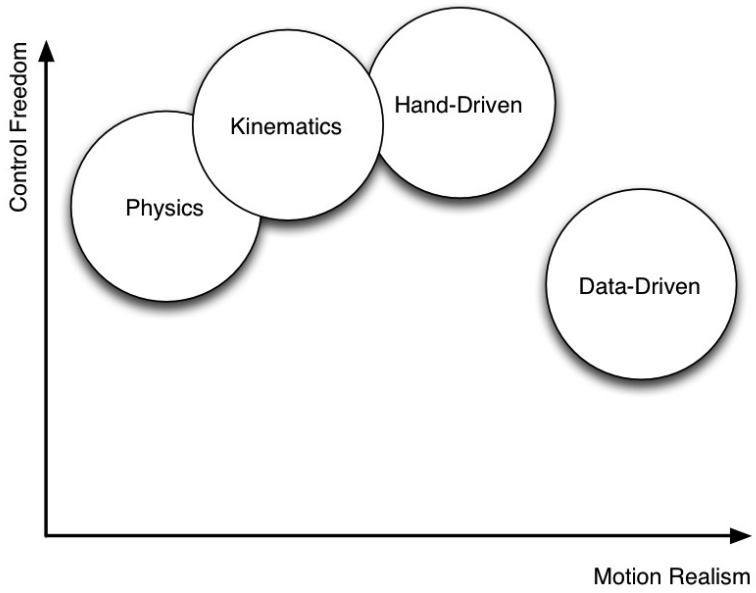

Figure 2.6: Evaluation of the animation methods between the control freedom (vertical axis) and the motion realism (horizontal axis).

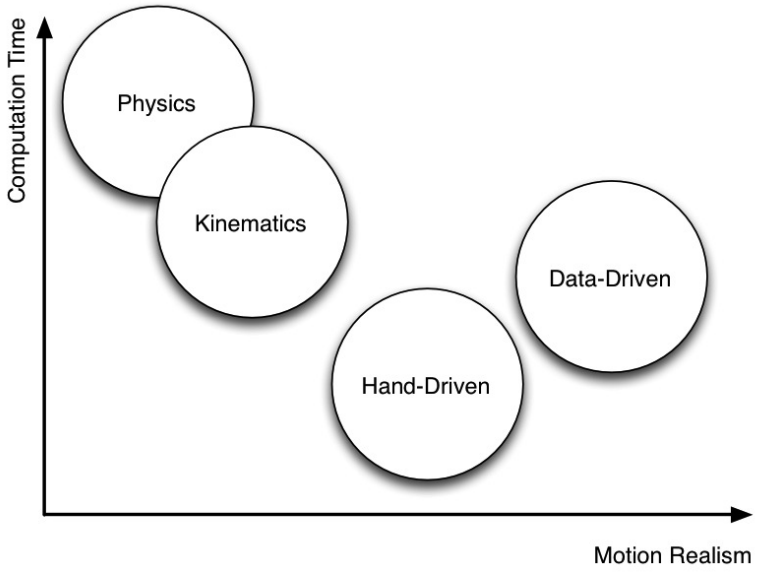

Figure 2.7: Evaluation of the animation methods between the computational time (vertical axis) and the motion realism (horizontal axis).

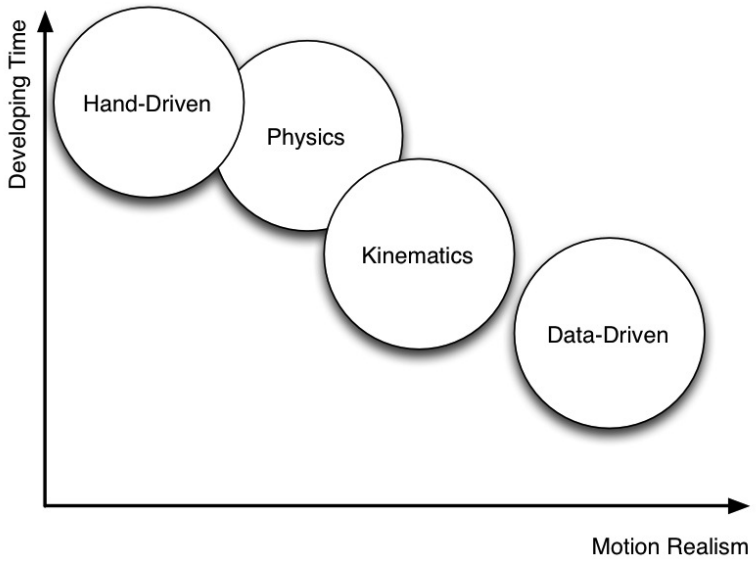

Figure 2.8: Evaluation of the developing time that is required by an experienced user for achieving the required realism of character's motion.

techniques for designing the necessary patterns through motion capture data Tanco and Hilton (2000) providing the ability of synthesising valid postures of the character. During the past years, a significant body of work had been produced on constructing generative statistical models for human motion analysis and synthesis. In general, the generative statistical motion models are often represented as a set of mathematical functions, which describe human movement using a small number of hidden parameters, such as Brand and Hertzmann (2000) Grochow et al. (2004), and their associated probability distributions. Thus far, a wide variety of generative statistical models have been developed and their applications include motion analysis and synthesis, inverse kinematics, and motion style interpolation and transfer.

Behind those methods, the ability of handling the motion data plays the most important role. Thus, firstly it is required being found such a data clustering method that will provide the ability of analysing in a correct manner the motion data. Basically, even if various methodologies for data clustering have been developed during the past years, the most well-known as well as the most widely-used in computer animation is the principal component analysis (PCA) Alexa and Müller (2000), which is responsible for reducing the dimensionality of the data by two thirds, and the variogram functions Mukai and Kuriyama (2005), which describes the degree of spatial dependence of a spatial random field or stochastic process. Hence, rather than analysing the motion data in the 3D space, it is possible to analyse the desired components in a reduced dimension space, such as the one provided by PCA (see Figure 2.9). In this case, it should be mentioned that

even if human's body does not move linearly, the majority of methodologies for human motion synthesis and editing uses linear motion clustering methods Carvalho et al. (2013) Carvalho et al. (2009).

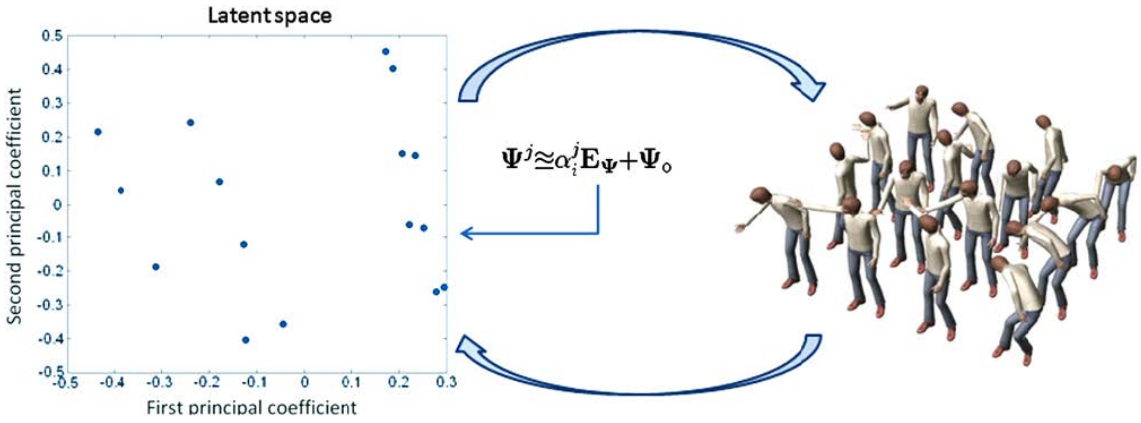

Figure 2.9: The representation of the end effector's position on the latent space by using PCA, as it is proposed by Carvalho et al. Carvalho et al. (2013). Each blue sample in the latent space represents a full-body motion in the joint space. This example shows a 2D mapping built from the rst and second principal coefcients.

Several other statistical models have been successfully used, including Hidden Markov Models (HMM) Brand and Hertzmann (2000) Tilmanne et al. (2012). Linear Statistical methods create a motion space using statistical models learned from the statistical variation of example motion primitives. Several statistical models can be used, including Independent Component Analysis (ICA) Cao et al. (2003), Gaussian Process Wang et al. (2007), Scaled Gaussian Process Latent Variable Models (SGPLMVM) Grochow et al. (2004), as well as various mixture models such as Mixture of Regressors Agarwal and Triggs (2005), Mixture of Factor Analysis (MFA) Lau et al. (2007), and Mixture of Probabilistic Component Analysis (MPPCA) Weise et al. (2011). Especially the mixture models are proved as the most powerful, since with those methods it is possible to synthesise new motion sequences without the need of having the exact similar motions.

It should be mentioned that those methods, firstly, provide a flexible way for analysing the existing data, as well as a flexible way for synthesising new motion sequences, where it is not required being located the exact motions or postures in the database. An additional advantage of those methods is the ability of approximating a valid, as well as a natural posture of the virtual character at each time step. Hence, since the ability of providing better results during the motion synthesis process, those methods attracted the research community. Finally, those methodologies are possible to run in interactive frame-rates, being able to synthesise or to reconstruct the desire motion during the runtime of the

applications.

2.4 Footprint-Driven Motion Planning

Footprint-based methods for synthesising the locomotion of the characters are quite efficient. This is since, comparing both with the motion graphs and the search trees, footprint-based methods are responsible for driving the character in the exact desired position in the 3D space, rather than in an approximation. Specifically, the footprints are placed interactively by the user/developed in the 3D space. Each footprint denotes for animating virtual characters based on footprints that have been proposed the desired position and orientation of the character's feet while it contacts with the ground. Thus, while it is ensured the exact position of each characters foot in the 3D space, it is possible the character to follow a constrained path reaching the exact desired position. It should be noted that footprint-driven methods for animating the locomotion of a character are potentially the best way to describe the positioning of the motion data in the 3D space.

Egges and Van Basten Egges and Van Basten (2010) proposed a data-driven method where a greedy nearest-neighbour approach warps the resulting animation to satisfy both spatial and temporal constraints. By using motion capture data as well, Wu et al. Wu et al. (2008) proposed a procedural solution, in which the trajectory of the foot and the Center of Mass (CoM) are determined, thereby enriching the resulting motion, in accordance with an optimization method that enforces the resulting CoM to follow the desired trajectory. Another solution that uses motion capture data to generate the desired steps based on footprints was proposed by Huang and Kallmann Huang and Kallmann (2010b), in which a pseudo-blending motion parameterisation process generates the desirable result. In this case, only the spatial position of the foot is achieved, rather than taking part in the motion synthesis process the exact foot positioning, as in the solution proposed by Van Basten et al. Van Basten et al. (2011), where a hybrid interpolation scheme based on both orientational and Cartesian interpolation was used for synthesizing exact foot positioning. Finally, Choi et al. Choi et al. (2003) proposed another solution that generates character's locomotion based on footprint. Specifically, in their method the virtual environment is sampled on possible footprint steps, constructing a roadmap. Then by augmenting the roadmap with a posture transition graph they traverse it to obtain the sequences of input motion clips.

On the other hand, physics and kinematics methods have been developed for animating virtual character based on footprints. Hence, Coros et al. Coros et al. (2008) developed

physical controllers that is aware of a footprint tries to follow it as closely as possible. Additionally, kinematics solutions proposed by Van de Panne Van de Panne (1997) and Chung and Hahn Chung and Hahn (1999b). Specifically, Van de Panne Van de Panne (1997) proposed a procedural space-time work, in which the CoM trajectory is determined based on a physical optimizer according to the inverse kinematics determining leg motions. In the approach of Chung and Hahn Chung and Hahn (1999b) the motion synthesis is generated by a hierarchical system on top of footprints laid over uneven terrain.

The aforementioned methodologies, especially these related to data-driven, are able to provide quite reasonable results, especially in the part related to the naturalness of the synthesised motion. Generally, each of the aforementioned methods has its own advantages and disadvantages. For example, as mentioned in Wu et al. (2008) the character is able to follow the footprints although, it is only ensured only the positional constraints of the foot. This limitation can easily be solved by enforcing kinematics constraints such as those used in Choi et al. (2003). Although, as kinematics are not always responsible for providing highly realistic posture of the character methodologies that are able to edit the motion data may be quite beneficial for solving the exact foot positioning problem. In Van Basten et al. (2011) the exact foot positioning solved by proposing a methodology, which allows the motion data to be edited by fulfilling both the position and orientation constrains. In the methodology presented in this thesis a novel technique for solving this problem is introduced. Specifically, it is examined the ability of synthesising the required motion based on an optimization problem of the motion blending process. The result over the presented methodology ensures that the character fulfills both the position and orientation constraints, as well as that the computational cost that is required does not influences its real-time implementation.

Conversely, even if editing the motion data for fulfilling the necessary constraints, there are other issues that may influence the naturalness of the synthesised motion. For example, a character should not only being responsible for performing only single actions, such as regular walking motion. It should be able to perform different variations of motion sequences. Thus, methodologies that allow the synthesis of different behaviours should be examined. Based on Glardon et al. (2005) the incorporation of real-human measurements can be quite beneficial in ensuring the naturalness of the synthesised motion while different behaviours evolved in a locomotion sequence. Although, in Glardon et al. (2005) it was examined only the ability of a character to perform three different actions such as the walking, running and jumping. Specifically, in the aforementioned methodology,

depending the target jump the character adjust its speed such as being able the system to provide a natural looking jump action of the character. A disadvantage of the aforementioned methodology is its inability to enure a human like transition between the origin and the target speed. Thus, the resulted synthesised motion looks unrealistic. For that reason, the methodology presented in this thesis deals with the natural transitions that should evolve while a different target behaviour is required being synthesised by the system. For achieving this natural transition existing motion data were analysed. This analysis was performed on a basis of footsteps. Hence, by knowing in advance the number of steps where a character requires achieving a target behaviour it is possible to adjust progressively the speed of the character allowing a smooth and natural looking transition.

Finally, the interactivity between the user and such a system should be flexible enough, such as enabling almost every user to interact with the system allowing those to synthesise the required motion quite easily. In previous works, especially in Wu et al. (2008) and Van Basten et al. (2011), the user places the footprints in the 3D space and the system is responsible to synthesise the motion of a character. Although, the aforementioned methodologies are not able to synthesise different behaviours of the character. Moreover, it is not clear enough in the aforementioned methodologies what is the resulted motion of the character while a footprint in not located in a reachable area. Hence, in the presented methodology by introducing the footprint patterns the system is able to recognise and to synthesise different behaviours of the character. Moreover, for avoiding misconceptions of the system, by introducing some simple computations the system is able to ensure that each footprint lies in the action space of the existing motion data contained in the database, by placing additional footprints automatically.

2.5 Performance Animation

With performance animation it is meant the procedure where the performer (user) is captured in real-time and its body parts are mapped to a virtual character. Therefore, the user manipulates the virtual character. In general, performance animation methods for synthesising or reconstructing the human motion rely on the ability of approximating a valid posture of the character during the motion capture process. The key issue where the research community focuses, is based upon the ability of approximating valid postures while the number of markers that are used for the motion capture process are reduced. Both the performance capture and animation methods are referred as “computer puppetry” Sturman (1998) Shin et al. (2001). Additionally, an article describing the usage of

natural interfaces for interactive character control had been written by Liu and Zordan Liu and Zordan (2011). Although, as the motion synthesis methods vary, the performance animation methods vary as well. Specifically, the methodologies for synthesising the human motion during the performance capture process are based on kinematics, on physics, as well as on data-driven methods. It should be noted that there are also hybrid methods such as Ishigaki et al. (2009) that combines physics and data-driven motion synthesis.

The use of kinematics for controlling virtual characters has been examined thoroughly. Badler et al. Badler et al. (1993) propose the use of four magnetic sensors and real-time inverse kinematics algorithms to control a standing figure in a virtual environment. This solution offers the ability to adopt a heuristic approach to handle the kinematic redundancy for a data-driven approach. Semwal et al. Semwal et al. (1998) proposed an analytic solution to the inverse kinematics algorithm based on eight magnetic sensors. Boulic and Raunhardt Boulic and Raunhardt (2009) used integrating analytic and linearised inverse kinematics to generate high-quality poses for a character. Finally, Unzueta et al. Unzueta et al. (2008) proposed a sequential inverse kinematics method for reconstructing the full-body posture of a character.

In the aforementioned methodologies the number of sensors that are used for capturing the performer, as well as the kinematics solutions may influence the naturalness of the synthesised motion. Generally, the greater number of sensors is used, the more realistic motion can be synthesised by the system. This is since, while the motion of the user is mapped to a virtual character, the kinematics solvers are not always able to describe the motion of the user. Hence, in cases where four sensors are used such as in Badler et al. (1993) the kinematics solutions should be much more constrained comparing with Semwal et al. (1998) Boulic and Raunhardt (2009) Unzueta et al. (2008) for providing a natural motion of a character. On the other hand, depending on the reason where the performance animation is used it is possible the system to synthesise natural motions only while using a few markers. A typical example is in Badler et al. (1993) where simple walking motions can be synthesised quite natural by using only four sensors. Although, more complex motions such as dancing cannot be synthesised with the required naturalness.

On the other hand, physics-based methods such as the one proposed by Ha et al. Ha et al. (2011) is responsible for approximating a natural pose of the character while a single force sensor is used for capturing the human motion. Other approaches that uses full-body motion capture, such as the solutions proposed by Nguyen et al. Nguyen et al. (2010), Ishigaki et al. Ishigaki et al. (2009), and Shum and Ho Shum and Ho (2012) provide the

ability of approximating a natural human positioning while the performer interacts with various tasks and objects located within the virtual environment. Finally, Shiratori and Hodgins Shiratori and Hodgins (2008) proposed a physics-based controller that allows the user to control the motion of a dynamically simulated, animated character through the motion of his or her arms, wrists, or legs.

It should be noted that the use of physics in performance animation in mainly used for enhancing the input motions, instead for synthesising the motions itself. Generally, physics methodologies can be used for synthesising the reactions of a character in external forces as in Nguyen et al. (2010) Shum and Ho (2012) or to approximate a physical valid posture of the character while it interacts with tasks where cannot be performed directly by the user as in Ishigaki et al. (2009). Specifically, in Ishigaki et al. (2009) the user through the character can interact with a monkey bar by extracting parameters related to the user's dynamics and by mapping these parameters to a physical motion model. Although, the synthesised motion, depending on how constrained such a model is, may provide or not a natural posture of the character. On the other hand, by computing parameters related to user's center of mass, forces and torques it is possible to synthesise physical valid postures of a character based on the analysis of the aforementioned parameters provided by existing motion data as in Ha et al. (2011) Shiratori and Hodgins (2008). Although, such motion synthesis techniques cannot be used for synthesising a variety of actions comparing with the kinematics methodologies mentioned previously.

Data-driven approaches for estimating the motion of a character have also been proposed as well. These approaches use existing motion sequences that are stored in a database. In a preprocessing stage before mapping the input signals, which have been retrieved from a motion capture device, to estimate a valid posture for the character, it is required a preprocessing stage for analysing the motion data, which is the most common approach for synthesising the motion of the character, such as those statistical models mentioned in the previous subsection. Based on data-driven techniques for human motion synthesis, various methodologies for handling the input signals and generating a valid motion for the character have been proposed. More specifically, Chai and Hodgins Chai and Hodgins (2005) proposed a system that automatically learns a series of local models from a set of motion capture examples that closely match the marker locations. This system uses six input signals retrieved from a motion capture device. Liu et al. Liu et al. (2011b) constructed a series of online local dynamic models from a pre-recorded motion database and utilised them to reconstruct full-body human motion. Kim et al. Kim et al. (2012)

proposed a method to reconstruct the motion of a character using kernel canonical correlation analysis methods to build a local model that transforms the low-dimensional input signals into a high-dimensional character pose. Finally, Liu et al. Liu et al. (2011a) used a mixture model with factor analysers to synthesise high quality results from six sensors used for approximating the posture of the character.

Generally, data-driven techniques are responsible for providing quite natural human postured during the motion synthesis process comparing both the kinematics and physics-based methods. This is since, the data-driven techniques uses existing motion data for animation the character. Although, the naturalness of the synthesised motion is always dependent both on the number of input parameters that are used as well as on the statistical motion model that is used for analysing and synthesising the motion of the character. Generally, the data-driven performance animation methodologies use six markers Chai and Hodgins (2005) Liu et al. (2011b) Kim et al. (2012) Liu et al. (2011a) for animating the virtual character. This number of markers seems to be the optimal since the reconstruction error is quite low. Although, the statistical motion models that are used as well as the number of data that is used affects this reconstruction error. For example, it is possible to synthesise quite natural motion sequences by using a large and heterogeneous motion capture database as in Liu et al. (2011b) Kim et al. (2012) Liu et al. (2011a). On the other hand, by using a small number of data that contains only similar motions it is not possible to animate different actions of a character as in Chai and Hodgins (2005).

By summarising the performance animation techniques, it can be stated that the naturalness of the synthesised motion is partial depended on the number of markers that are used during the motion capture process as well as on the constraints where the developed applied for synthesising the motion of a character. Specifically, depended on the kinematics solver or on the motion model that are used for the motion analysis process, the synthesised motion may lie or not within the natural looking space. Finally, it should be mentioned that the advantage of performance animation is its usage in various fields such as in virtual reality and video games, in rehabilitation, as well as in sports sciences for analysing the naturalness of the human's motion.

Based on the aforementioned explanation of the different techniques that are used for the performance animation process in this thesis a data-driven methodology that synthesises the style variations of the user during the performance capture process is introduced. The presented techniques uses a statistical motion model, the Mixture of Probabilistic Principal Component Analysis for handling the motion data contained in a database. This

motion model was chosen since it reduced the reconstruction error, as indicated in Weise et al. (2011), while a small number of motion sequences are used, especially compering with the one used in Chai and Hodgins (2005) Liu et al. (2011b) Liu et al. (2011a).

2.6 Style Motion Synthesis

With style motion synthesis it is referred the methodologies that are able to compute the necessary style variations of a motion sequence, and then to transfer this variation to another motion sequence. A typical example is a motion sequence of a stylistic behaviour, such as a happy walking motions that is required being transferred in a normal running motion. Thus, by extracting the necessary features that characterise the stylistic behaviour, those features can be mapped to any other motion that is required to fulfil the style behaviour.

Therefore, while the ability to edit or synthesise new motion sequences by keeping the style variations of existing sequences is required, methodologies that are related to transferring the motion styles of one motion to any other have been previously proposed. The parameterisation process for the motion data can be quite powerful in cases that require the prediction of a new motion style from existing motion data. In general, dimensionality reduction techniques, such as principal component analysis (PCA) and Gaussian process latent variable models (GPLVMs), can be quite beneficial in cases that require a classification process to discriminate between different motion styles. More specifically, methodologies that are capable of synthesising style variations from human motions are presented below.

Urtasun et al. Urtasun et al. (2004) used PCA to train a large motion capture dataset with variations on locomotion style sequences. Using PCA coefficients, they synthesised new motions with different heights and speeds. Brand and Hertzmann Brand and Hertzmann (2000) used Markov models to capture the style of training motion, which were archived to synthesise new motion sequences while the style variations of the motion were transferred. Torresani et al. Torresani et al. (2007) proposed a motion blending-based controller, where the blending weights were learned from a large dataset of motion sequences in which the motions' styles had been labelled by specialists. Liu et al. Liu et al. (2005) constructed a physical model using an optimisation approach to generate motions with learned physical parameters that contained the style aspects. Another solution proposed by Elgammal and Lee Elgammal and Lee (2004) assigned style properties to time-invariant parameters and used a decomposable generative model that explicitly

decomposed the style from a walking motion video. Moreover, independent component analysis (ICA) has been widely used for synthesising motion sequences with stylistic variations. Cao et al. Cao et al. (2003) used ICA to automatically determine the emotional aspects of facial motions to edit the styles of motion data. In addition, Shapiro et al. Shapiro et al. (2006) used ICA to decompose each single motion into components, providing the ability to select the style component manually.

Hsu et al. Hsu et al. (2005) proposed a style transfer methodology that learns linear time-invariant models by comparing the input and output motions to perform style translation. Unuma et al. Unuma et al. (1995) proposed a method that uses Fourier techniques to change the style of human gaits in a Fourier domain. Using this method, the motion characteristics based on the Fourier data could be extracted. Bruderlin and Williams Bruderlin and Williams (1995) edited the stylistic motions by varying the frequency bands of the signal. Perlin Perlin (1995) added rhythmic and stochastic noise functions to a character's skeletal joints to synthesise motion sequences with personality variations. Finally, Neff and Fiume Neff and Fiume (2006) proposed a methodology in conjunction with a prototype system to generate stylistic character animations. This solution provides animators with a set of edit modes and bridges the gap between artistic and technical communities.

Based on the aforementioned methodologies a general pipeline of the motion style transfer can be stated. Generally, in a first step, the style content that distinguish two motion sequences is computed according to various methodologies. This is achieved in off-line computations. Then, having computing the necessary style variations, the system is able to transfer the extracted style to a different motion sequence, which is a process that can either be provided in real-time or off-line. The advantage provided by the style motion synthesis techniques is the ability of avoiding the repeated capture of specific stylistic content of different behaviours for enhancing a collection of motion data. Thus, it is possible by having a limited number of stylistic behaviours, to compute the necessary content and then to transfer this content to any other motion.

2.7 Finger Motion Synthesis

In previous years, a number of different approaches have been proposed for synthesising finger motion sequences by simplifying the rules of the motion synthesis process. Methods that automatically synthesise hand motions, known as hand-over animation, have been proposed. In general, the hand-over animation process relies on the ability to add finger

and hand motion capture data to the pre-existing full-body motion capture data. Hence, solutions, such as those proposed by Kang et al. Kang et al. (2012), and Wheatland et al. Wheatland et al. (2013), are responsible for adding the required motions of the character's hand even when a reduced number of markers are used for the motion estimation process.

The first approach to introduce the automatic addition of valid motion of the character's finger was proposed by Jin and Hahn Jin and Hahn (2005). In this approach, the pose space distance from the character's motion capture data is analysed and a stepwise searching algorithm is used to find the key poses the synthesised hand motion. Building on the benefits of the previous solution, Jörg et al. Jörg et al. (2012a) proposed one of the most interesting solutions for finger motion synthesis for gesturing characters. This approach is based on the ability to synthesise a character's finger motions by assigning weight variables to the wrist's position and orientation, which is used as the input information for the control parameters of the motion synthesis process. This solution searches for short segments of the wrists motions in a database to determine an optimal path by taking advantage of the motion graph literature Kovar et al. (2002a).

Although, even if the aforementioned methodology provides the ability of synthesising hight quality finger motions, the basic disadvantage lie on the ability of estimating all the required motions correctly. This is since, their system uses only spatial information provided by the character's hand. Hence, comparing this solution with the approach presented in this thesis, it is achieved to estimate in a better manner almost every gesture of the character. This is achieved by introducing a hybrid methodology that computes and analyse motion features of the character's motion. Based on the motion features, the system is able firstly to classify the motion segments into gesture and non-gesture phases. The gesture phases are then classified in similar gesture types. This classification methodology provides a reduction of the actual computational time where the system requires to estimate the most valid motion. Moreover, due to the classification process it is achieved a generalised representation of each gesture type. Hence, by searching through the generalised representation of the motion segments it is possible to estimate in a better manner every gesture of a character. Both the reduction of the computational time, and the growth of the estimation rate are the main advantages of the presented methodology.

Another solution for synthesising a character's finger motion was proposed by Majkowska et al. Majkowska et al. (2006). In this methodology, finger motion and body motion are captured separately in a pre-processing stage. Then, during the composition process, those motion sequences are assembled using spatial and temporal alignment meth-

odologies, and the motion correlation is found. In general, this technique takes advantage of motion transplantation techniques Ikemoto and Forsyth (2004) Van Basten and Egges (2012) Ha and Han (2008). These methods have been previously used to combine different body parts' motions to a new motion sequence to attempt to maintain a natural final generated motion.

The approaches for synthesising a character's hand motion are always constrained to specific tasks. This is due to the difficulty of predicting a valid human finger motion. Even if using a whole-body motion it is difficult to predict the exact actions of the hand, which are highly correlated with the fingers. Hence, other solutions, such as the one proposed by Ye and Liu Ye and Liu (2012), are responsible for generating the motion of the finger based on wrist movements and the handled object's specified motion constraints. Specific manipulation strategies are assigned to the fingers, while the character's wrists are used to predict the evolution of a specified action.

Similarly, finger motion synthesis solutions for specified tasks have been previously developed, such as solutions where virtual characters interact with music instruments and the motions of the characters are automatically generated. One of the most recent approaches proposed by Zhu et al. Zhu et al. (2013) assigns a specific action to the fingers in conjunction with the ability to execute information from a MIDI input and from predefined parameters of the piano performance. Their system generates a valid motion sequence for the virtual character, where the fingers play an active role in the motion synthesis process. Similarly, the solution proposed by ElKoura and Singh ElKoura and Singh (2003) generates finger motion for specific tasks, such as musical instruments.

Other works use sensors to measure forces Kry and Pai (2006) to attempt to generate the correct finger motion. Moreover, Neff and Seidel Neff and Seidel (2006) derived synthesised human hand motions by using relaxed hand shapes and physics based parameters retrieved from video recordings. Finally, another interesting solution for animating detailed and anatomically valid hand and finger motion was proposed by Tsang et al. Tsang et al. (2005). Although, the main disadvantage of all previously mentioned solutions is the computation cost that is required to generate the desired motion of the fingers.

In general, the data-driven techniques Jörg et al. (2012a) are able to provide quite natural looking motions of the character's fingers since the motion data is resulted from real humans. Although, two basic disadvantage characterise those methods that is the time consuming computations and the estimation rate of the correct finger motion. Especially for the finger motion estimation process, methodologies that automatically estimate the

most probable motion of the character is able to provide the desirable results, although, the wrong estimation rate may affect negatively the perception of the observer. Thus, it is assumed that methodologies that are able to estimate in a more valid manner a wide range of finger motions while a reduced number of input parameters are used could be quite beneficial. On the other hand, hand over animation techniques Kang et al. (2012) Wheatland et al. (2013) can provide in a better manner a realistic as well as a correct motion of the character's fingers. This is since, while the input parameters increased it is increased the estimation rate of the estimation process.

Conversely, while synthesising the motion of the characters finger based on kinematics solvers, the synthesised motion may not looks naturals, due to the kinematics are not always able to provide the necessary naturalness. Although, for specific tasks such as in the solution proposed Zhu et al. (2013) where only simple motions are desired, the system is able to synthesised high quality motions of the character's fingers. Finally, physics based methodologies Tsang et al. (2005) can be quite beneficial for the finger motion synthesis, especially in cases where interaction between the fingers and objects are required. Although, these techniques suffer from the required high computation making the aforementioned methodologies quite difficult to work in real-time.

2.8 Discussion

In this chapter, related work on character animation was presented. As mentioned, there are three basic computational categories that can be used for animating a virtual character: the kinematics, the physics and the data-driven. Moreover, combination of those techniques such as combining kinematics with physics or kinematics with data-driven have also been developed. Those hybrid techniques are in general responsible for solving particular problems in computer animation such as the undesired foot sliding effect. Additionally, each of these techniques has each own advantages and disadvantages. Thus, based on a simple evaluation it was shown that the basic advantages of the data-driven techniques is the required realism where an animated character must have, as well as minimum rates of the computational time that is required, which allows the real-time implementation of such techniques.

It was also presented related work on the different areas in computer animation where the presented research focuses. Specifically, it was presented related work on statistical motion analysis and synthesis, on performance animation, and on style motion synthesis, which have been employed for developing a methodology for rea-time performance-based

animation with style variation, on finger motion synthesis and on locomotion synthesis. Based on the related work on this research, it should be noted that there are certain disadvantages or misconceptions. Therefore, in the following sections, novel methodologies for either improving or proposing a different way for solving particular disadvantages are presented.

Chapter 3

Locomotion Composition

One of the most efficient ways of generating goal-directed walking motions is synthesising the final motion based on footprints, as discussed in Section 2.4. Current implementations have not examined the generation of continuous motion based on footprints, where different behaviours can be generated automatically. In this chapter a flexible approach for footprint-driven locomotion composition is presented. The presented solution is based on the ability to generate footprint-driven locomotion, with flexible features like jumping, running, and stair stepping. In addition, the presented system examines the ability of generating the desired motion of the character based on predefined footprint patterns that determine which behaviour should be performed. It is also examined the generation of transition patterns based on the velocity of the root and the number of footsteps required to achieve the target behaviour smoothly and naturally. Finally, it should be noted that the motion data used in the presented methodology captured by real humans performing different locomotion behaviours.

3.1 Introduction

During the past few years, the generation of motion sequences based on foot placement has been examined intensely Egges and Van Basten (2010) Wu et al. (2008) Huang and Kallmann (2010b) Van Basten et al. (2011) Choi et al. (2003). This approach is important in cases where natural locomotion and collision avoidance is required, especially in highly constrained environments, such as indoors. In addition, since characters should be able to perform locomotion especially for goal-directed tasks, footprint-driven motion synthesis allows the generation of long motion sequences, in which the character can reach the exact desired position rather than an approximation. However, footprint-driven motion

synthesis should not necessarily be limited to single actions, such as walking or running, as a vast number of applications related to virtual reality, such as video games, requires footprint-driven locomotion composition to involve multiple motion types.

The advantage of the presented solution, compared to existing footprint-driven motion synthesis techniques is the ability to generate a character's locomotion, rather than simple walking motions. This is achieved by extracting features related to footprint patterns to determine the different actions of the character during the composition, as well as by generating transition patterns based on the velocity of the character while another action is evolving. In the presented solution, motion variations based on jumping, running, and stair stepping are presented to enhance the generated motion sequence. When dealing with different motions it is necessary to examine the generation of long motion sequences that retain the characteristics of each individual motion while allowing smooth transition between the different motion variations. Hence, the presented system automatically generates the desirable motion variations based on precomputed properties of each motion and then disseminates these to the footprints resulting in smooth and continuous motion.

3.2 Overview

The motion synthesis process is split in to the three main components shown in the following subsections. These components are the ability to handle the motion capture data (Section 3.2.1), the motion extraction process (Section 3.2.2), and the motion blending process (Section 3.2.3) that is responsible for placing the foot in the exact required position.

3.2.1 Motion Handling

For each segmented motion the foot that performs the action is denoted as p_{act}, and the foot that supports the action as p_{sup}. Specifically, p_{sup} is the foot that remains in contact with the ground during a single step, while the p_{act} is the one that performs the step. In addition, for the foot performing the action a semantic label is assigned describing the starting and final position, such as p_{start}, which is the starting position, and p_{end}, which is the final position. Each supporting foot p_{sup} is represented with the global position of foot joint, its $y - axis$ orientation, and the global position of the toe joint, such as $p_{sup} = (x^b, y^b, z^b, \theta^b, x^t, y^t, z^t)$ the positions of p_{act} are represented in the same way but maintaining the local position, rather than the global, of the foot joint and toe joint, based on p_{sup}, that is, $p_{start} = (x_1^b, y_1^b, z_1^b, \theta_1^b, x_1^t, y_1^t, z_1^t)$ and $p_{end} = (x_2^b, y_2^b, z_2^b, \theta_2^b, x_2^t, y_2^t, z_2^t)$. Therefore, each motion contained in the database can be

described as $m = \{p_{sup}, p_{start}, p_{end}, v_{root}, \xi\}$ where v_{root} and ξ are the velocity of the root and the semantic label of the motion respectively. Figure 3.1 depicts the parameters of a footstep.

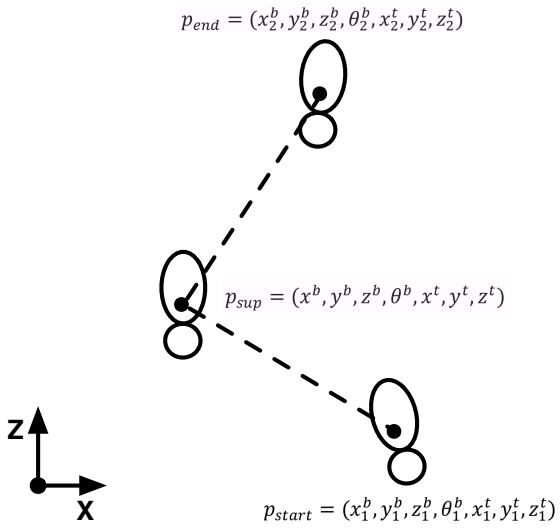

Figure 3.1: The parameters assigned for each step.

For the extraction process, first all the supporting foot motions p_{sup} must be aligned to the desired position and orientation of the footprint placed in the three dimensional space. This can be done by aligning the corresponding vectors. Then, the position of the supporting foot is represented as $f(p_{sup})$. Given p_{start} and p_{end} footprints, the next step is to extract the most suitable motions thereof for blending. Finally, it should be mentioned that the reason that the p_{act} is assigned these parameters is the ability to extract and parameterise separately the reference motions of the foot joint and the toe joint. Then, those motions are assembled in order to provide the desirable result, as discussed in the following subsection.

3.2.2 Motion Extraction

For the motion extraction process, firstly each position of p_{start} and p_{end} with global orientation θ_i are sorted with the orientation of the desired reference position. Then, considering that $\theta_1 \leq \theta_i \leq \theta_n$, where θ_n is the $n-th$ registered orientation of the example motions, it is necessary to extract the most suitable of these, so as both the position and the orientation of the footprint lie within the desired position. For this reason, two example motion sequences are always considered, where $\theta_l \leq \theta_n$ must be satisfied for the first one, and $\theta_k \geq \theta_n$ for the second. Then, two more motions sequences are selected following the same procedure. Each footprint should be enclosed by two polygons (see Figure 3.2) that

are produced based on the positions p_{end} of the reference motion data, where each edge of the generated polygon corresponds to a p_{end} reference position. Each suitable p_{end} is used for the blending process. If multiple combinations of p_{end} can satisfy this requirement, the polygon enclosing the foot joint, with the minimum sum of the distances is used for blending, thereby providing the desired result. The same process is used for extracting the most suitable motions that enclose the toe joint. It should be mentioned that the chosen motions sequences must fulfill both the positional and orientation criteria. This results in the ability to synthesise the exact foot placement.

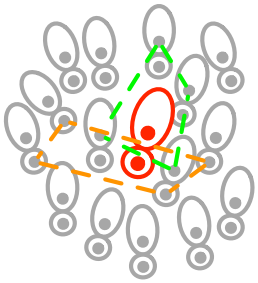

Figure 3.2: The polygons that enclose the foot joint (orange) and toe joint (green) are responsible for the blending process. The grey footprints denote the reference target motion sequences, and the red footprint a defined footprint.

3.2.3 Motion Blending

In this case, by directly blending the reference motions of the foot joint based on the form $P^f_{start} = \sum_{i=0}^{3} w_i \times P_i$, where P^f_{start} is the desired position of the foot joint, $\sum_{i=0}^{3} w_i = 1$, and P_i is the corresponding position of reference motion m_i, it is possible to retrieve the desired spatial position of the foot joint without ensuring the desired orientation. Then, using the same procedure, the most suitable motions that enclose the toe joint are blended in order to provide the desirable result.

For the motion blending process, having retrieved the blend weights w_i ensuring the positional constraint of the foot joint and the toe joint, the next step is the recalculation of the weighted interpolation function of these for each degree of freedom separately, allowing the orientation constraints to be retrieved. In this case, the exact foot positioning can be computed as:

$$P(p_{foot}, p_{toe}) = v_1 \times P_{foot} + v_2 P_{toe} \tag{3.1}$$

where v_1 and v_2 are the blendweight variable, P_{foot} and P_{toe} the position of the foot joint and the toe joint based on the resulted blending motion sequences. Although, since the

direct blending does not ensures the exact foot positioning, the blendweights v_1 and v_2 are computed based in the following optimisation problem:

$$P(p_{foot}, p_{toe}) = \min \sum_{i=0}^{S} \|v_1 \times P_{i,foot} + v_2 \times P_{i,toe} - P_i\| \tag{3.2}$$

where S denotes the total number of motion ($S = 8$, four for the foot and four for the toe) that are used for this blending process, P_i is the exact foot position that is required being used for the blending procrss, and $\sum_{i=1}^{2} v_i = 1$. Now, for solving this optimisation problem, the least square method was used. Thus, v_1 and v_2 are estimated as follows:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} P_{1,foot} & P_{1,toe} \\ ... & ... \\ P_{N,foot} & P_{N,toe} \\ 1 & 1 \end{bmatrix}^{+} \cdot \begin{bmatrix} P_{fp} \\ ... \\ P_{fp} \\ 1 \end{bmatrix} \tag{3.3}$$

where $[]^{+}$ denotes the pseudo inverse of a matrix.

Using this approach, the character can satisfy more precisely the defined spatial and orientation constraints, as illustrated in Figure 3.3. Finally, it should be mentioned that during the blending process footsliding effect may appear. Hence, the technique proposed by Kovar et al. Kovar et al. (2002b) that enforces an inverse kinematics solver for removing this undesired effect was enforced.

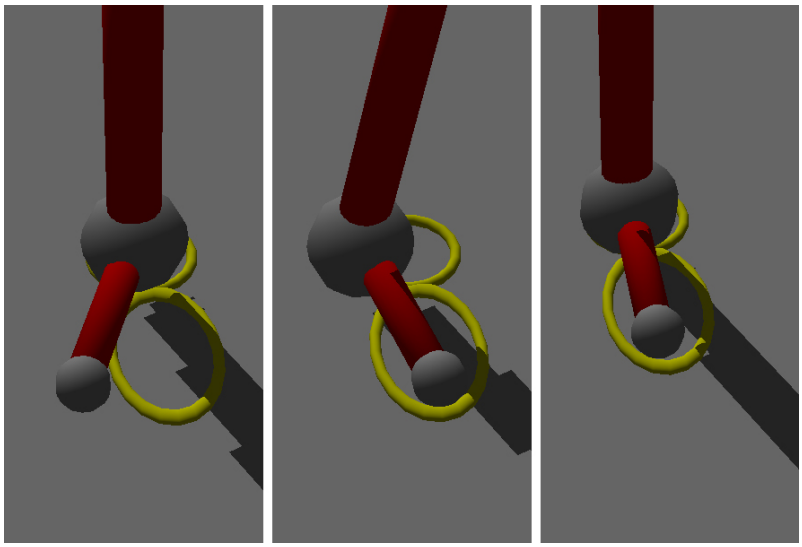

Figure 3.3: The spatial alignment of the foot based on the presented approach. The alignment of the foot joint (left), the alignment of the toe joint (middle) and the final generated alignment (right).

3.3 Locomotion Composition

In the following subsections, the ability of synthesising the locomotion of the character is presented. More specifically, it is examined the ability of assigning the required behaviour of the character in the footprint patterns, and the ability of the transition process between different behaviours in the action transition graphs.

3.3.1 Footprint Patterns

In a simple footprint-driven walking sequence the character is able to perform the desired walking motion based on the actual motion sequences located in the database. However, while generating locomotion the actual step space can be extended. Hence, it is necessary for the system to recognise which locomotion should be generated based on the footprints placed in the three dimensional space. In general, changing the locomotion behaviour from walking to stair stepping is considered easy because it is possible to retrieve the foot placement based on its height. On the other hand, since the character should be able to exhibit more complicated behaviour, such as jumping, which could involve both feet jumping simultaneously or jumping by lifting one foot first, the system must be able to recognise which action matches the user inputs. Thus, even if the semantic labels of both actions are similar, the system should generate the most suitable motion.

In the presented approach, footprint patterns are generated to compute the relative position of the origin and target footprints. Thus, the footprint patterns are divided into two categories, namely, global footprint patterns, which are responsible for recognizing which action best matches the behaviours of walking, running and jumping, and local footprint patterns that are generated for jumping actions, so that the system can recognize which jumping action is the best match. As illustrated in Figure 3.4, for the global patterns among the three different behaviours, it was examined the distance between the two states of the action foot. Specifically, the plane (x, z axies) distance between the two stances of p_{act} is measured automatically by the system in a preprocessing stage by analysing the collection of motion clips for each particular behaviour. The maximum distance denotes the higher limit of the particular behaviour. Finally, it should be mentioned that when these limitations are based on a database of motion, as well as in cases where humans can perform the same motion with different variations, this approach cannot be generalised. Nevertheless, it can be used as a parameter to handle the motion capture data and generate the desired motions.

The approach for generating local behaviours is quite important, especially when a

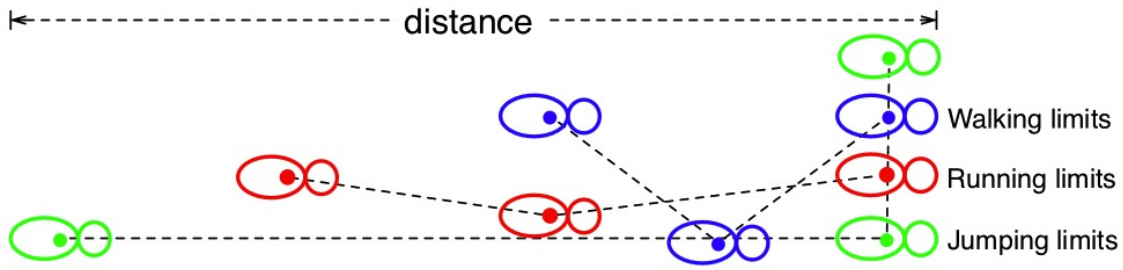

Figure 3.4: The global footprint patterns generated for walking, running and jumping behaviours.

jumping action needs to be performed, ensuring that the generated motion satisfies both the origin and the target footprints. Thus, four different local footprint patterns are generated (see Figure 3.5) allowing the system to automatically generate the desired jumping action that should be carried out. Specifically, for the generation process of the local footprint patterns, the system automatically analyses the angles θ_i and θ_j (as those angles are illustrated in Figure 3.6) between the feet positions both in lifting and in landing processes. This averaged angle is used as the threshold for determining the particular jumping action of the character. Thus, the character can perform jumping actions with either one or both feet lifting simultaneously, as well as with either one foot or both feet landing simultaneously. Finally, actions (b), (c), and (d) illustrated in Figure 3.5 are also mirrored for the other foot.

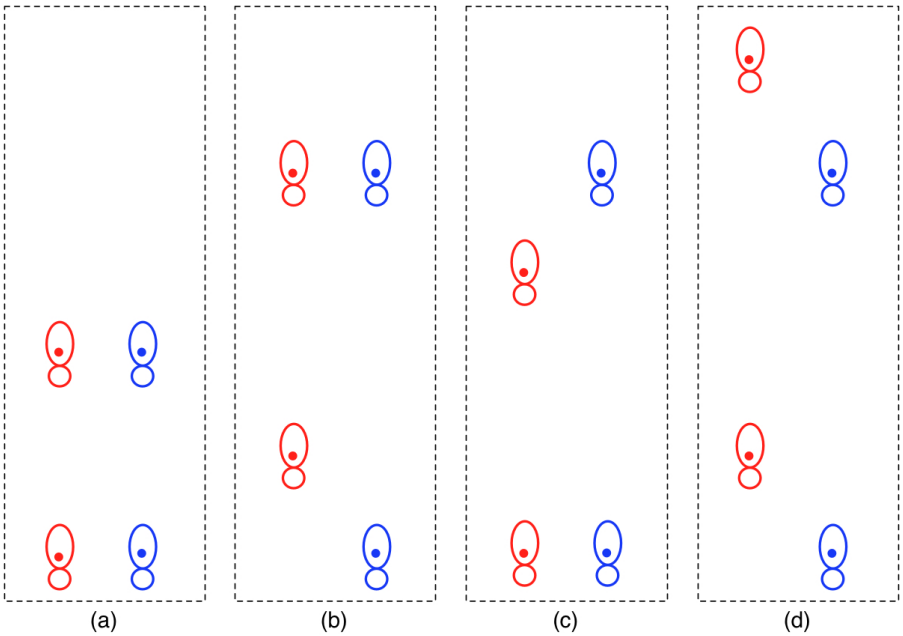

Figure 3.5: The four different footprint patterns used to generate jumping actions.

For the extraction process of walking, running and jumping, it is measured the distance that the active foot travels during a single step. Using this distance the system determines

which behaviour to perform. If the extracted motion is either walking or running, the system directly builds the desired step. On the other hand, if the distance is within the limits of a jumping action, the system searches for the jumping behaviour that best matches the footprints. In this case, the generated angle θ between p_{act} and p_{sup} the original position, as well as at the target position, in conjunction to the distance between the p_{sup} and p_{act} are examined. A threshold angle, $\theta_{thres} \in [-\theta^o, \theta^o]$, which results from the jumping motion analysis process and a distance $d_i \leq d_{max}$ (see Figure 3.6) are used to generate the pattern in which both feet jump and/or land simultaneously.

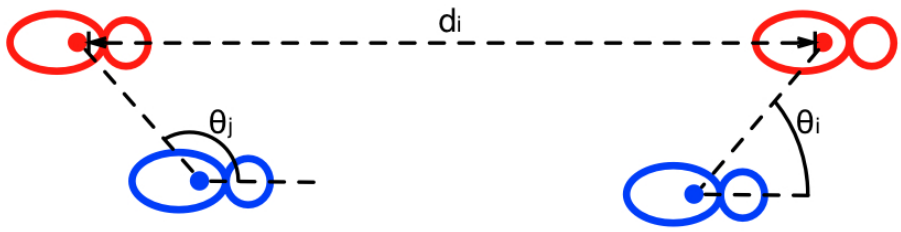

Figure 3.6: The parameters of the pattern in which both feet jump and/or land simultaneously.

To represent jumping actions, each pattern is assigned as $\Pi = \{\theta_i, \theta_j, d_i\}$, where θ_i and θ_j are the angles between p_{sup} and p_{act} at the lifting and landing position of the jumping action respectively. The height of any foot is not computed since the methodology deals with the traveled distance where each singe step provides. However, the height of the foot could be used for describing additional locomotion behaviours. Depending on the actual value, the system measures one or both feet lifting or landing. d_i denotes the actual distance between the p_{start} and p_{end} position of character's foot p_{act} and it is used in order to extract jumping actions, in which the character travels a certain distance. Although, since during the jumping action there is not any foot contact with the ground the supporting foot does not exists. In this case, it should be mentioned that the system assigns as supporting foot the one which is used before the lifting process, as well as assigns as supporting foot the one which is first landed. It should be mentioned that the footprint patterns are common related with the existing motion sequences. Hence, in cases where a character with different height is calling to perform the desired locomotion, undesired actions may be synthesised by the system. An example where characters having a different high is illustrated in Figure 3.7.

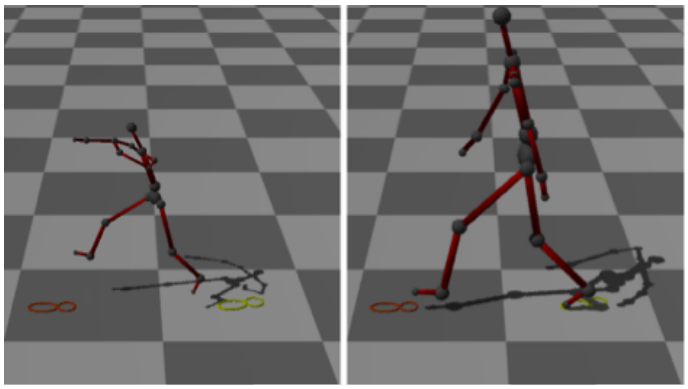

Figure 3.7: Characters with different high performs different motions while using the same footprints.

3.3.2 Behaviour Transition

This section examines the transition process that was used in the presented implementation. This transition process provides to the synthesised motion toe ability to appear as natural as possible. As observed in the experiments, unnatural motions are generated especially when the character changes behaviour from, for example, running to stair stepping, since the velocity of the character changes dramatically. For that reason, it was considered a process that progressively changes the velocity of the character during a behaviour transition process.

For achieving a smooth transition from one behaviour to another, a velocity-based approach for the root trajectory is used. To understand how each behaviour influences the generated motion sequence, in a preprocessing stage, the velocity of the character's root is measured while the character transits from one action to another. The results are semantically clustered based on the different behaviour transitions, linearly mapped according to the footsteps and the average number of footsteps required to proceed from one behaviour to any other is measured in conjunction with the velocity of the character's root. Transition graphs do not have to be generated for each combination of motions because motions that have a related velocity transit correctly, compared to those that the velocity changes dramatically, such as changing from walking to running actions. Hence, in the presented approach five different transition graphs are generated and used, as shown in Figure 3.8. Mirrored versions of these graphs are used to generate the inverse transition process. For retrieving those transition graphs the following steps, as those presented in the following subsections, were carried out.

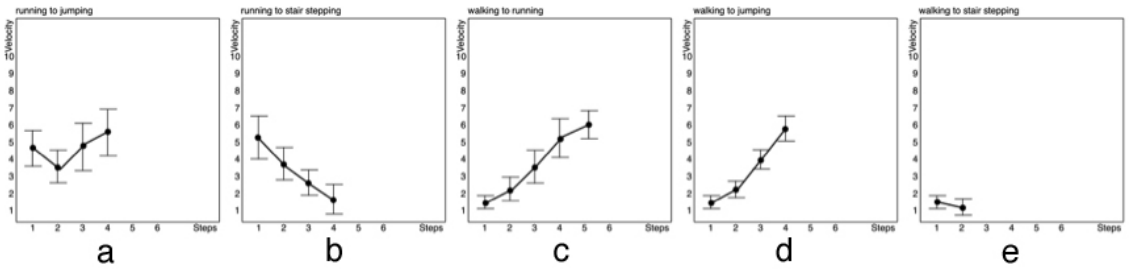

Figure 3.8: Transition graphs based on different transition behaviours: (a) from running to jumping (both feet jumping), (b) from running to stair stepping, (c) from walking to running, (d) from walking to jumping (one foot lifting and landing first), and (e) from walking to stair stepping. The horizontal axis shows the required footprints, while the vertical axis shows the velocity of the root in (m/s).

Transition Identification

For each motion, the velocity of the root, which characterises the transition process, was assigned as the main parameter for understanding the differences between the individual actions employed in the motion sequence. More specifically, the velocity of the root between two different actions should have a minimum or a maximum value that characterises one of the target actions. This approach was used to generate the desired velocity components assigned to each transition (see Table 3.1). For example, in the transition from a walking to running motion, the inverse transition, from running to walking, is executing by inversing the registered results, rather than using measurements from the motion capture data. The inverse process is used because it is assumed that this procedure can provide the desired result, as well as reduce the number of generated transition graphs.

From \ To	Walking	Running	Jumping	Stair Stepping
Walking	-	max	max	min
Running	-	-	max	min

Table 3.1: Discrete velocity characteristic used for the transition process between different actions. Velocity components with (-) were not computed.

Based on Table 3.1, it is possible to generate the velocity that characterises each target action. However, as the velocity is not the sole parameter, the number of steps taken to move from the origin to the target is also measured. This measurement is made using

a simple foot detector Van Basten et al. (2010), which recognises foot contact with the ground based on the velocity and height of the foot. Using this approach, the velocity component that characterises the target action is determined, as is the time period for which the velocity of the action is maximised or minimised. It is then possible to determine the number of steps required to generate the velocity representing the maximum or minimum value.

Transition Alignment

Having calculated the root velocity in accordance with the foot contact approach, the next step of the method is to align the motions. The alignment process is based on the ability to align the number of steps with the root velocity. More specifically, two features are extracted for each motion: the number of steps s_i and the velocity of the root v_i at each step, such that each recorded motion m_i is represented by $m_i = \{(s_1, v_1), ..., (s_n, v_n)\}$, where n denotes the $n-th$ registered step of the human derived from the motion capture data. Having extracted each action transition from those components, the next step is to align the motions (see Figure 3.9) based on the $i-th$ step, where the $i-th$ root velocity is equal to the maximum velocity v_{max} or the minimum velocity v_{min}, depending on the discrete behaviour of the action transition mentioned above. This alignment ensures that all of the registered motions have the same transition behaviour and are aligned with the correct phase.

Transition Graphs

The presented transition graphs represent the velocity of the root of each motion, as based on the corresponding steps. Considering that the number of steps for each registered motion varies, depending on the transition process, it is necessary to identify which steps should actually be included in the graphs. For the motion analysis process, it is necessary to remove the undesired steps (i.e., those that are not taking part of the transition process). Thus, for computing the actual steps that are required, the mean average of the number of steps before and after the velocity component is used to validate the number of required steps for the transition process.

Having aligned the steps based on the velocity property and having removed the undesired steps, to generate correct transition graphs, the actual numbers of steps and the root velocity are plotted. Based on the graph, it is possible to develop a general approach to the transition between two different actions by measuring the mean root velocity at

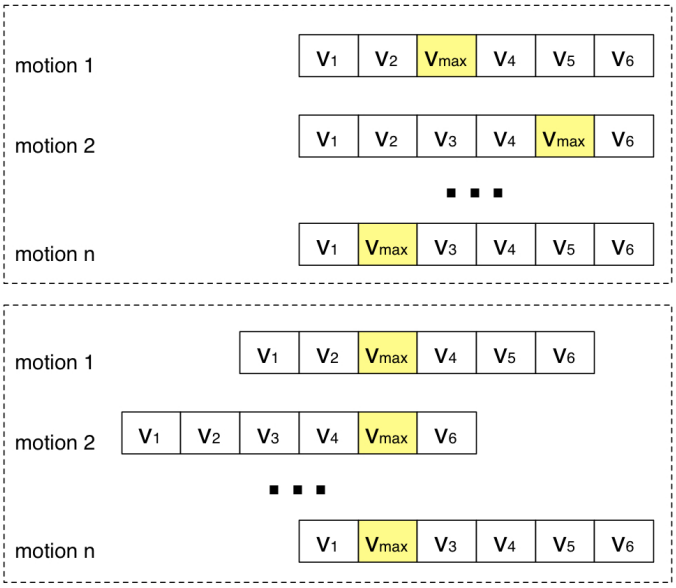

Figure 3.9: Alignment process of the step for which the velocity of the root is assigned to the velocity component that characterises the target action. An example is given for before the alignment (upper plate) and after the alignment (lower plate). Each rectangle denotes a step; the yellow rectangles denote the step at which the velocity component is executed.

each step. Based on the mean average of each action transition at each step, an action transition graph is constructed which presents the mean velocity of the root at each step of the character, presented as $m_{action} = \{(s_1, v_1^{mean}), ..., (s_n, v_n^{mean})\}$. With this representation of the transition process, it is possible to understand how each action transition evolves.

3.4 Ensuring Continuity

To ensure the continuity of the movement, the system extracts information on a number of previous and subsequent footprints to determine which action should be performed. In the previous section generating motion that transits smoothly from one behaviour to another is discussed. The same metrics are used when a footprint is located in an area where the limits of two motions intersect in order to determine whether it denotes a long step or a running action. According to Figure 3.8(c) and considering that the original action is walking, the number of steps required by the character to generate a running action is four. Thus, by measuring only the previous footprints, the system cannot provide the exact desired motion since the subsequent footprints determine how the motion should evolve.

To enable the system to determine whether a running action should be performed, the

next four steps are measured. Then, if all four steps are located within the intersection area of walking and running ($A_{walk} \cap A_{run}$), all of these steps are considered to be long steps. On the other hand, if at least one of those steps is located in the running action area only, the footprints are interpreted as a running action. In addition for each action the following limitation placed. The lower limit of the running actions is the upper limit of the walking actions. Similarly, the lower limit of the jumping actions is the upper limit of the running action. Hence, even if intersection between two actions exists, it is possible to extract the correct action of the character by using the distance limitation in conjunction with the information retrieved from the action transition graphs.

3.5 Implementation and Results

This section discusses the implementation of the presented solution, the system functionalities, and example motion sequences generated with the presented methodology. For the implementation of the presented solution, 100 steps of simple walking actions, 50 steps retrieved from running actions, 100 jumping actions (25 actions for each jumping condition) and 50 stair stepping actions were used. Mirrored versions were constructed for all the aforementioned motions, resulting in a total number of 600 motions. All motions were downsampled at 60 fps. Finally, the motion synthesis process were performed on an Intel i7 2.2 GHz processor with 8 GB RAM.

3.5.1 Implementation and Results

Although the generated actions of a character are strictly limited to those contained in the database. A certain correction of those actions was necessary in order to avoid incorrect computations that are based on the ability of synthesising the locomotion of the character based on the motion handling approach. Depending on the action performed, two different approaches are implemented to correct the continuity. If a footprint is located higher in the three dimensional space than that supported by the registered motions in the database, the position of the supporting footprint p_{sup} is changed by:

$$f(p_{sup}) = f(p_{sup}) + l_{stair}^{height} - f(p_{end}) \tag{3.4}$$

where l_{stair}^{height} is the upper limit of stair stepping motions. When the supporting footprint is also higher than the upper limit, the procedure that changes the height of the previous footprint continues iteratively backwards to the previous footprints. In this way, the approach provides an easy method of generating multiple stair steps without aligning all

the footprints. On the other hand, automatic adjustments can be made in case a footprint is placed outside the limits of jumping actions, which have higher limits based on the global footprint patterns. In this case, the system divides the displacement between the end footprint $f(p_{end})$ and the upper limit for the previous footprints, that is:

$$f(p_i^{prev}) = f(p_i^{prev}) + \frac{|f(l_{action}^{height}) - f(p_{end})|}{n_{prev}} \tag{3.5}$$

where $f(p_i^{prev})$ is position of any previous footprint, $f(l_{action}^{height})$ is the position of the upper limit of the action used, and n_{prev} is the total number of previous footprints. The function $f(\cdot)$ is used for describing the global position of the footprint. Although, this small adjustment can also be done in an arbitrarily large number of previous footprints. It was chosen to apply some simple rules, enabling the system to interpret easily any wrong position of a footprint.

By combining the two approaches, it is possible to enhance the synthesis process, avoid wrong estimation (e.g. a wrong estimation may occur while a footprint does not describe a particular action that the system could generate) and assist the user in understanding to what extent each footprint influences the rest of the motion. Furthermore, the division process also works for stair stepping actions: if all previous footprints are deadlocked -that is all footprints are adjusted to higher limits- a new footprint is generated automatically on the top of the last one. With this approach it is ensured that always all the footprints that take part to the locomotion composition will be able to provide a valid motion. This is since all the footprints stay within the actual action space of motion where the character performs. It should be mentioned that imposing limitations directly on the footprints could circumvent these procedures, thereby avoiding footprints placed outside the boundary limits.

3.5.2 Results

For the completion of the computation process, each step required 0.189 seconds on average (based on both generated locomotion illustrated in Figure 3.12). The computational time required per step was divided between two different processes. Motion parameterisation, and motion blending, took 39%. The motion synthesis process based on motion graphs Kovar et al. (2002a) used the remaining computation time, that is, 61%. The computational cost of the method may seem high, but given the complexity of the problem and compared to other solutions presented in the literature, it undoubtedly represents an improvement to existing systems. Figure 3.10 illustrates the linkage between the number

of motion sequences that were used, in conjunction with the ability of synthesizing each single step of the character. Considering the computational cost of the solution proposed by Egges and Van Basten Egges and Van Basten (2010) (where they use an Intel Pentium Centrino 2.4 GHz, and 200 motion sequences stored in their database), a single step requires 3ms computational time on average. The computational time of the presented approach, for 200 motion sequence (100 walking, 50 running, 30 stair stepping, and 20 jumping) was estimated to be 0.076 seconds. The cost difference between the solution proposed by Egges and Van Basten Egges and Van Basten (2010) was anticipated, since in the presented approach additional parameters related to the motion extraction process, such as action transition and exact foot positioning, influence the motion synthesis process.

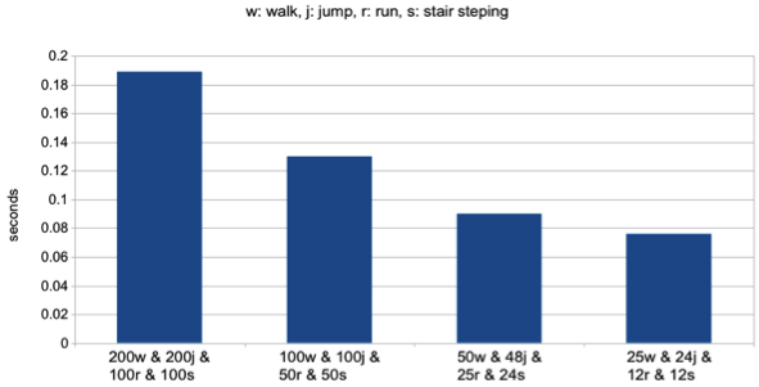

Figure 3.10: The time required for the generation of a single step, based on different database size.

The innovation of the presented solution is the ability to infer and synthesize the correct behaviour of the character based on the defined footprints. Specifically, the system is able to generate jumping actions based on the patterns presented earlier. Figure 3.11 shows the generation of different jumping actions based on the placement of the footprints. In addition, the ability of the locomotion composition process to generate long motion sequences with the footprints located across a desired path, which can be either straight or curved, was tested. Figure 3.12 illustrates example motions generated based on the different paths. It was also examined the ability of the presented approach to generate different behaviours during the locomotion. Figure 3.13 depicts two long motion sequences in which all the examined behaviours are implemented. It should be mentioned that even if a large collection of motion sequences is used to generate the motion of the virtual character, there are still motions that the database does not contain. In this case, the movement generated does not appear to be completely natural, since the motion blending technique cannot always provide the desired result.

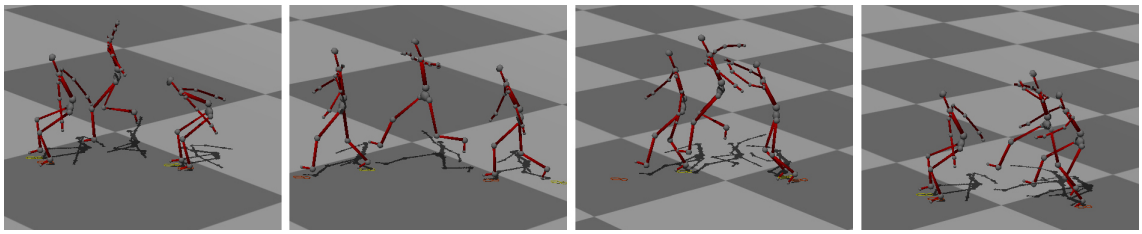

Figure 3.11: Generated jumping motions based on different footprint patterns.

Figure 3.12: Generated motion based on a straight (left) and curvature (right) path.

3.6 Discussion

In this chapter a solution for automatically synthesizing continuous locomotion of a virtual character based on footprints was presented. In particular, it is showed that the presented solution could generate a vast number of different actions, such as running and jumping actions. Since the system is able to recognise the desired actions of the character, the presented solution adds the ability of synthesising the desired action of the character by assigning the motion synthesis process to footprint patterns, enabling the identification of each action. In addition, especially in common related actions, such as the jumping actions, local footprint patterns developed to determine which motion is the best match. Additionally, it is often desirable to generate not only continuous motion based on different behaviours, but also continuous motion in which the different behaviours can transit smoothly, it has been shown that generating motion transition patterns related to the velocity of the root in accordance with the required number of footprints result in a smooth transition from one behaviour to another. Although, it should be mentioned that the effectiveness of the method is highly depended on the number of motion sequences. Thus, the more variety of reference motion sequences, the more variety of behaviours of the character that can be generated.

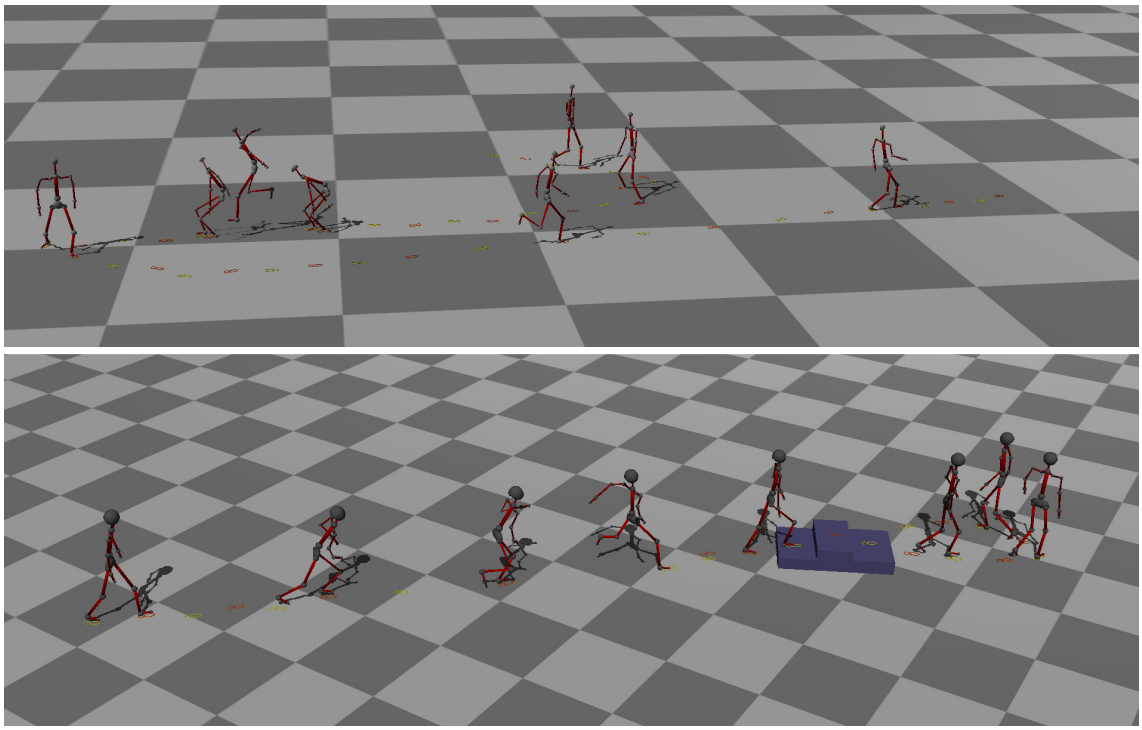

Figure 3.13: Different behaviours implemented in a single locomotion procedure. Example motions that consist of 46 footprints with walking, sidestep, running and jumping actions (upper row), and another generated locomotion procedure that consists of 24 steps with walking, running, jumping and stairs stepping actions (lower row).

Chapter 4

Synthesising Style Variations

Virtual characters are desired to be able to animate by using different style behaviours. With style behaviours it is meant the different perspective where real humans, or in this case the virtual character, perform a specific motion. For example, a walking motion can be either a neutral, or happy or sad and so on. Therefore, any variation of a motion can be considered as style behaviour of the neutral motion. Even if in the previous solution it was examined the ability of synthesising a single style variation of the character's locomotion, which can be characterised as a neutral motion, in this chapter it is presented a simple methodology for synthesising in real-time motion sequences of virtual characters with style variations during the performance capture process. The approach builds a simple correlation between two individuals performing the same motion with different style variations. Separate movements are captured for each performer. These movements are then synchronised using a dynamic time warping technique. Using a radial basis function (RBF), the relationship between the two motion sequences can be defined. During the application runtime, a MAP framework is used to achieve the desired real-time performance for the motion synthesis process. The presented method has practical applications in films and games that rely on performance capture. Rather than using a vast number of actors who each perform a specified motion with a specified style according to the virtual character, this techniques makes it possible to synthesise the desired style variations using only a few actors performing neutral motions. A simple explanation is given in Figure 4.1. Specifically, using a correlation process it is possible to find the mapping between two different style behaviours. Thus, during the synthesis process, a different actor is able perform its neutral movements and the system to synthesise a stylistic perspective of the input motion.

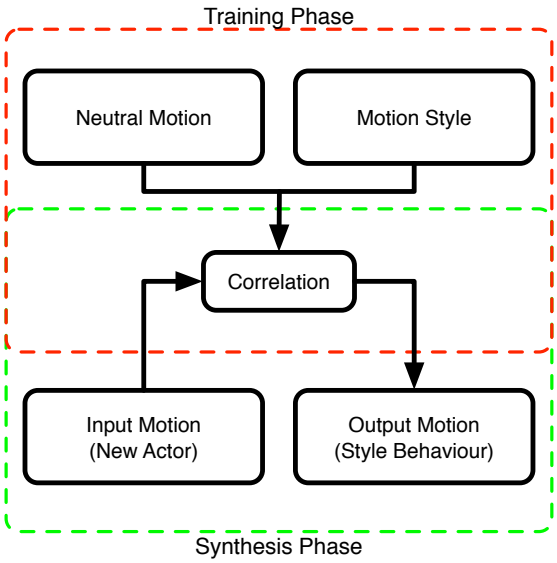

Figure 4.1: A simple explanation on how the presented methodology works.

4.1 Introduction

In film productions, the capture process for each individual character requires the ability to capture various actors embodying the desired roles and styles. The use of multiple actors for human motion capturing cannot be easily achieved in small-scale or amateur productions, though it can be achieved efficiently in major productions Osipa (2010). Based on this limitation, there is a need to design a methodology in which the different style variations of human motions can be mapped to a performer's motion to generate the desired motion sequences for different stylistic behaviours in real-time. The key advantages of such a solution would be the ability to (i) build a correlation between two different motion styles (a normal motion and a stylistic representation of the motion) and (ii) generalise the motion model to be able to maintain the stylistic variations of the motion in cases required for a new stylistic motion that is not present in the motion database. To achieve a correlation between two different motion sequences, the RBF can be used. The mixture of probabilistic principal component analysis (MPPCA) method can be used to achieve a generalisation of the existing motion data in conjunction with a maximum a posteriori (MAP) framework. Thus, the RBF can be used as a parameter over the likelihood function of the MAP to synthesise a desired stylistic representation of the user's performance in real time.

4.2 Methodology

The methodology of the presented approach is based on the following steps. Firstly, two different actors are captured performing the same motion (e.g., walking). Each actor performs the motion with a different style variation. Secondly, the motions are synchronised using a dynamic time warping methodology. Third, the two different motion sequences are correlated. Finally, the desired style is synthesised in real-time based on the MAP framework. The pipeline of the presented methodology is illustrated in Figure 4.2.

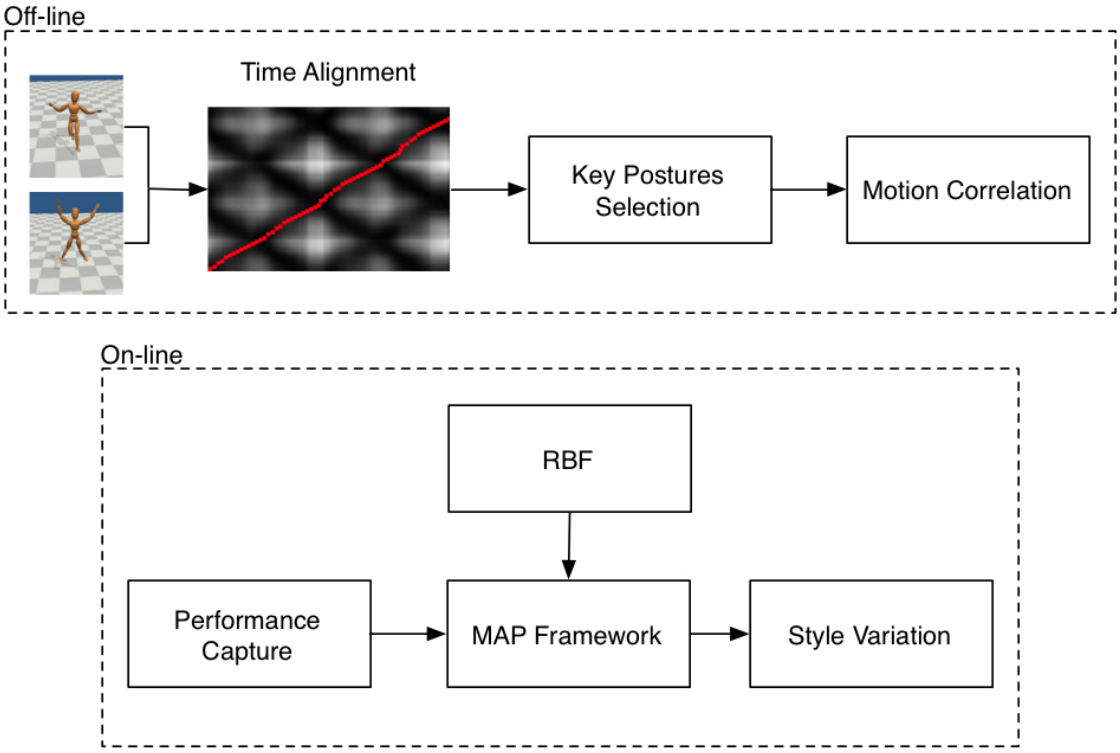

Figure 4.2: Pipeline of the presented methodology.

4.3 Motion Preprocessing

The preprocessing stage of the approach is divided into three major steps. First, the corresponding pairs of captured motion sequences are synchronised with respect to time using a time warping methodology. Any undesired frames are removed to prevent interference with the synchronisation. Once synchronised, a simple search method finds the time step in which the style variation is maximised. This is done for every captured motion of the specified stylistic motion. Finally, having found the corresponding postures, the motion sequences are mapped to the corresponding poses of the character to build a linear correlation. Those preprocessing steps are presented below.

4.3.1 Motion Synchronisation

The first preprocessing step aligns and synchronises the motion sequences. The motion sequences are aligned in pairs with respect to the motion time. Unnecessary frames that do not correspond to any motion are removed. Therefore, a time warping method is responsible for the synchronisation process that generates the time alignment curve $c(t)$, which is a strictly increased function that maps the frames of the normal M_N to the frames of the style M_S motion. In this case, the time alignment curve $c(t)$ is constructed using the M_N motion as a reference motion sequence. More specifically, the methodology is based on the work proposed by Kovar and Gleicher Kovar and Gleicher (2003). It was chosen this method since it provides an easy way for synchronising the motion data. Moreover, it provides the ability of finding directly the corresponding time periods that two motions align, providing the ability of easily removing the undesired time periods that do not match.

During the time alignment process, stylistic locomotion and non-locomotion sequences are aligned with the corresponding normal motion sequences. Thus, different body parts are used to approximate the optimal time alignment of the distance metric. Specifically, when dealing with locomotion sequences, for each frame, a point cloud is generated based on the positions of the character's root, feet, and hands. Conversely, when dealing with non-locomotion sequences, the point cloud is formed between the character's root and hands. In both cases, a distance metric is used to compute the optimal curve, where the sum of the distance between every corresponding frame along the curve is the minimum. More specifically, let $d(a, b)$ be the distance between the motion M_N at the frame a and the motion M_S at frame b. The time alignment process is required to determine a time alignment curve $c(t)$, where $c(t) = (a(t), b(t))$ for $t = 1, ..., T$ and T denotes the length of the curve. Therefore, to compute a time alignment curve between every corresponding frame, the following distance metric is used:

$$D(M_N, M_S) = \min \sum_{t=1}^{T} c(t) \tag{4.1}$$

Finally, having computed the distances between every corresponding frame pair, it is possible to draw a distance image where the time alignment curve is computed over the dark areas that denote the shorter distance (see Figure 4.3). This is achieved by assigning the results provided by $D(M_N, M_S)$ to greyscale value. Thus, a greater distance between M_N and M_S denote a greater value on the greyscale (0-255, where 0 denotes the black and 255 the white colour). Finally, in cases where the initial frames M_N and M_S are not in phase, an appropriate number of frames are cropped from the beginning of each

motion. The cropping approach is achieved by computing the distance between $M_N(t_0)$ and $M_S(t_0)$ at the first local minimum.

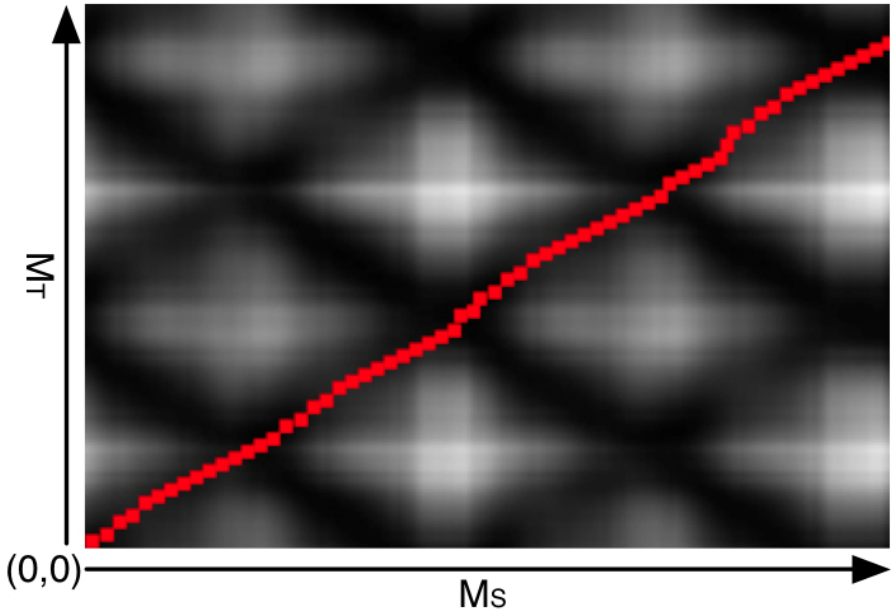

Figure 4.3: The time alignment curve (red line) represents the synchronisation process between the motion sequences. Dark areas denote the similarities between the corresponding frames.

4.3.2 Selecting Key Postures

In general, each motion sequence is represented by a number of individual poses of the character, such as $M_N^i = \{P_1, ..., P_i\}$, and $M_S^i = \{P_1, ..., P_j\}$, where i and j denotes the $i-th$ and $j-th$ posture of the character respectively. In addition, each pose of the character is represented by $P = \{p, q, r_1, ...r_m\}$, where p and q represents the root position and orientation of the character, respectively, and r_m represents the orientation of each $m-th$ joint of the character. Therefore, first the character's root position and orientation must be aligned, for each pair of corresponding motions M_N^i and M_S^i, and then to find the key poses that represents the style variations.

Those key poses, correspond in the time periods t_{max}, and t_{min} where each character's joint at both the M_N^i and M_S^i synchronised motions have its maximum and minimum difference respectively. The requirement of computing the time period of the maximum difference between two motions is based upon the ability to find the actual stylistic difference between two motions. A simple example that is illustrated in Figure 4.4 shows the maximum difference between two postures. Although, since this maximum difference should correspond at the whole-body of each character, the squared distance between the

two motions for each joint of the character at each time step is computed. Hence, firstly the root position of both motions is removed, and then the motion M_N^i is aligned (position and orientation alignment) to the corresponding motion M_S^i. It should be mentioned that for each pair of motion 48 poses are computed (24 for t_{max} and 24 for t_{min}) as the number of bones of the character according to the Acclaim Skeleton File (ASF) format, retrieved from the CMU motion capture database Carnegie Mellon University (2014) (see Section 4.5.1).

Having found those postures, it is now computed the correlation between the poses that represent the stylistic variation of a motion as it is represented in the following subsection. It should be mentioned that this step could be replaced by directly capturing the required key postures that represent the normal and the style variations. Although, it was chosen to present the searching process for the key postures since the presented motion synthesis process requires the style motion data for enriching a motion collection for the prior learning process, as it is presented in Section 4.4.3.

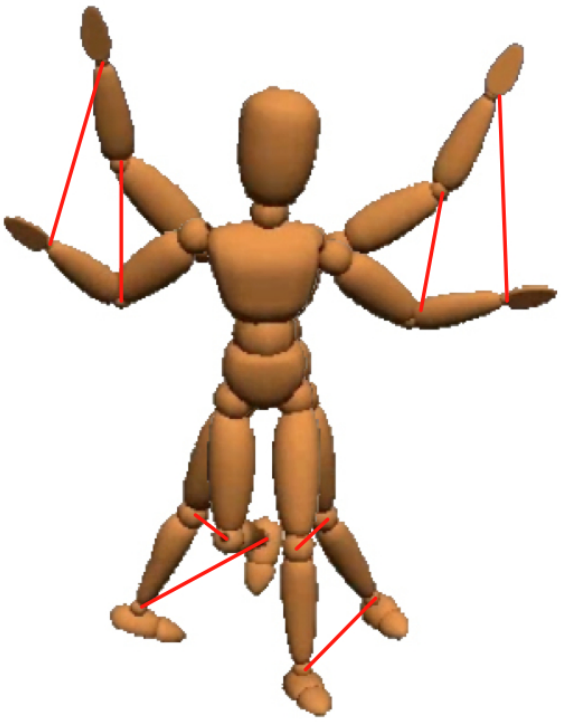

Figure 4.4: The two aligned motions. Lines between the joints show the maximum difference between its positions.

4.3.3 Building the Motion Correlation

Having synchronised, as well as having found the optimal style variations of each pair of the motion sequences, the correlation between every two motion sequences should be

built between M_N and M_S. Therefor, for the correlation process, the RBF is used for each pair of synchronised motion sequences, which is a powerful interpolation method Noh and Neumann (2001) Jung and Na (2004). It has been widely used for scattered data interpolation problems, such as surface approximations and fluid simulations Szécsi and Szirmay-Kalos (2010). In the presented approach, RBF is used to determine the best weights for the source and a target pairs of motion sequences. Mathematically, the RBF is represented as:

$$\varphi(x_i) = \sum_{j=1}^{N} w_j \times e_j(x_i) \tag{4.2}$$

where e_j is the radial function of the $j-th$ component (its value depends on the distance between the target and the origin pairs of the synchronised human postures), w_j denotes the computed RBF weights, N denotes the number of the range of style pairs (i.e., the actual number of human poses after the key poses founding process, see Section 4.3.2), x_i denotes the input vector, and $\varphi(x_i)$ is the estimated output.

Though several commonly used radial functions have been proposed, in the present approach it was chosen to be used the Hardy multi-quadrics function, which is represented by:

$$e_j(x_i) = \sqrt{(\|x_i - x_j\| - d_j^2)} \tag{4.3}$$

where d_j denotes the measured distance between x_i and its nearest x_j, leading to as little deformation as possible to the widely scattered feature points and as much deformation as possible for the closely located points. It should be noted that the Hardy multi-quadratic function was chosen since it provides the ability for handling unevenly distributed data sites with greater consistency that previous methodologies as described in Hardy (1990).

To find the correspondences between the pairs of motion sequences, it is necessary to train the RBF network. Specifically, for training the RBF network two steps are required. In the first step, the centre vector x_j of the RBF functions in the hidden layer are chosen. In general, this step can be performed in several ways such as by using $k-mean$ clustering or by randomly sampling the centres. In the presented methodology the training was performed by using the $k-mean$ clustering. It should be notes that this step is unsupervised. In the second step, a linear model with coefficients w_i fits to the hidden layer outputs with respect to some objective function. A general overview of the RBF function and its training process is given in Buhmann (2003).

After the training process, a linear operator $K = [w_1, ..., w_i, ..., w_N]$ is obtained, where w_i is the $i-th$ $N-$dimensional weight vector. Finally, by exploiting K, it is possible to

find an output vector, which is represented mathematically as follows:

$$v_{output} = K^T \cdot e(d_{input}) \tag{4.4}$$

It should be noted that RBF-based regression models cannot outperform in the case of non-linearly corresponding data samples. When an animator constructs the range of expression pairs, if two source expressions that are almost identical are mapped to the significantly different target expressions, it will cause conflicts in training step Deng et al. (2006).

4.4 On-line Style Motion Synthesis

During the on-line motion synthesis process, an actor performs the motion that is desired for the character. In this case, the on-line motion synthesis process is responsible for enhancing the motion of the actor by adding the desired style variation. In the presented solution, a small number of correlated motion sequences are used. This solution is based on the ability to generalise the existing stylistic motion data (e.g., synthesising the desired stylistic motion, even if it is not contained in the database).

To avoid this issue, the motion synthesis process uses a mixture model the MPPCA. In general, the mixture model is responsible for analysing the existing motion data in conjunction with a MAP framework that synthesises the desired motion in real-time while maintaining the desired level of naturalness for the existing motion data. It should be mentioned that the correlation between the normal and stylistic motion sequences is used as a parameter for the MAP framework, allowing the system to directly synthesise the desired stylistic motion of the character.

4.4.1 Data-Driven Style Motion Synthesis

The motion synthesis process is formulated in a MAP framework, which handles the input data retrieved from the motion capture sensor, generating the map for the correlation procedure presented in the previous section (Section 4.3.3) and assigning the input correlation to the likelihood function. The prior function is responsible for constraining the motion to be as close as possible to the training examples.

More specifically, the motion capture sensor automatically measures the global location and orientation of each body part of the performer $[c_1, ..., c_t]$ in real time. The system then reconstructs the stylistic approach q^* for the current user's pose control signals c_t

obtained from the motion capture device by parameterising the input signals with the stylistic approximation of the user's pose s_t.

More specifically, let c_t be the input data retrieved from the motion capture device that is required to reconstruct the stylistic motion s_t at the current $t-th$ frame. By combining the current control signals s_t provided by the motion capture device and the constructed probabilistic model from the previous m values of the reconstructed poses $\tilde{Q} = [\tilde{q}_{t-1}, ..., \tilde{q}_{t-m}]$, the system is able to reconstruct the current pose q^* in a constrained MAP framework, represented by:

$$q^* = \arg\max_{q_t} p(q_t|s_t, \tilde{Q}) \tag{4.5}$$

where $p(\cdot|\cdot)$ denotes the conditional probability. Using Bayes' rule, it is obtained the following:

$$q^* = \arg\max_{q_t} p(s_t|q_t, \tilde{Q})p(q_t, \tilde{Q}) \tag{4.6}$$

Assuming q_t is conditionally independent from $\tilde{Q}$, given s_t it is obtained the following:

$$q^* \approx \arg\max_{q_t} p(s_t|q_t)p(q_t, \tilde{Q}) \tag{4.7}$$

In this case, by applying the negative logarithm to the posterior distribution function $p(q_t|s_t, \tilde{Q})$, it is possible to convert the constrained MAP problem into an energy minimisation problem represented by:

$$q^* = \arg\min_{q_t} \underbrace{-\ln p(s_t|q_t)}_{E_{likelihood}} + \underbrace{-\ln p(q_t, \tilde{Q})}_{E_{prior}} \tag{4.8}$$

where the first term ($E_{likelihood}$) is the likelihood term that measures how well the reconstructed pose q_t of the character fits the stylistic pose s_t. The likelihood term is the parameterised term from the correlation process between the stylistic and normal motions. The prior (E_{prior}) is the term that describes the prior distribution of the human motion data.

4.4.2 Likelihood Distribution

The likelihood distribution process is modelled based on the correlation process between the normal and stylistic motion sequences. According to Equation 4.4, the system retrieves the current poses of the character s_t, which are represented as follows:

$$s_t = K^T \cdot e(c_t) \tag{4.9}$$

In this case, if the Gaussian noise with standard deviation σ_d is considered, the likelihood term is defined as:

$$\begin{aligned} E_{likelihood} &= -\ln p(s_t|q_t) \\ &\propto \frac{\|f(q_t;\tilde{g},\tilde{z}) - s_t\|^2}{2\sigma_d^2} \\ &\propto \frac{\|f(q_t;\tilde{g},\tilde{z}) - K^T \cdot e(c_t)\|^2}{2\sigma_d^2} \end{aligned} \tag{4.10}$$

where the vector q_t represents the reconstructed pose of the character at each time step, $\tilde{g}$ denotes the skeleton model, $\tilde{z}$ denotes the coordinates of each input pose of the character retrieved from the motion capture device, and c_t are the observed data obtained from the defined input pose. Finally, function $f(\cdot)$ is a forward kinematics function that maps the current pose q_t to the global coordinates.

4.4.3 Modelling the Motion Priors

In the presented methodology, the prior term for the statistical motion model is represented by the MPPCA model Tipping and Bishop (1999a). More specifically, the probabilistic principal component analysis (PPCA) Tipping and Bishop (1999b) is a parametric statistical model that defines the probability density function of some observed data $q \in R^D$, assuming q is a linear function for a latent variable $z \in R^d$ with $D > d$ such that:

$$q = Lz + \mu + \varepsilon \tag{4.11}$$

where $z \sim N(0, I)$ is distributed according to a Gaussian unit, $L \in R^{D\times d}$ is the matrix of the principal components, μ is the mean vector, and $\varepsilon \sim N(0, \sigma^2 I)$ is a Gaussian-distributed noise variable. Hence, the probability density of q is represented as follows:

$$p(q) = N(q|\mu, LL^T + \sigma^2 I) \tag{4.12}$$

The main goal of PPCA is to find the parameters of the model (μ, L, and σ), where the parameters must model as well as possible the covariance structure of the input data. The variable z models the correlations between each component of q, and the variable ε accounts for the independent noise in each component q.

Prior Distribution

Based on Equation 4.12, the prior distribution in Equation 4.8 is defined as a weighted combination of K Gaussian units, represented as:

$$p(q_t, \tilde{Q}) = \sum_{k=1}^{K} \pi_k N(q_t, \tilde{Q}|\mu_k, L_k L_k^T + \sigma_k^2 I) \tag{4.13}$$

with weights π_k. This representation can be interpreted as a reduced-dimension Gaussian mixture model that attempts to model the high-dimensional animation data with locally linear manifolds modelled by PPCA.

Learning the Prior

Now, it is desired the ability of the system to automatically learn the unknown parameters of the mixture model presented in Equation 4.13. Those parameters are the means μ_k, the covariance matrices $L_k L_k^T$, the noise parameters σ_k, and the relative weights π_k of each PPCA in the mixture model. In this case, the most common approach for learning these parameters is based upon using the Expectation Maximisation (EM) algorithm defined from the given motion sequences Verbeek (2004) Bishop (2006) Fraley and Raftery (2002).

More specifically, the goal of the model learning process is to automatically find the model parameters σ_k, π_k, μ_k, L_k for $k = 1, ..., K$ from the input training data q_n, $n = 1, ..., N$, where N denotes the number of poses contained in the database. Hence, the EM algorithm for MPPCA is adopted for learning all those parameters. It should be mentioned that since the EM algorithm converges to local minima, the algorithm run $K = 70$ times with random initialization to improve the learning accuracy. In addition $d = 5$ Gaussians used to model the prior. More details on the implementation of the EM algorithm can be found in McLachlan and Krishnan McLachlan and Krishnan (2007). It should be noted that the values for both K and d placed based on empirical evidence. Alternatively, cross-validation techniques could also be used to set appropriate values for these parameters.

4.5 Implementation Details and Experimentations

This section presents the implementation details for the style motion synthesis process, and the results obtained from different experimentations.

4.5.1 Implementation

To synthesise the stylistic perspective of the input motion data retrieved from the motion capture device, different stylistic motions are captured in conjunction with regular motion sequences. In the capturing process for the required motion sequences, the Microsoft Kinect Microsoft (2014) motion capture device was used, with its latest software that builds upon Shotton et al. Shotton et al. (2011). Some examples of the stylistic poses used in conjunction with the initial captured poses of the user are illustrated in Figure

Figure 4.5: The system retrieves the input poses (left) of the user (middle) and synthesises the stylistic variations (right) of the input motion.

4.5. In this case, it should be mentioned that all of the processing was performed on a 2.2 GHz Intel i7 with 8GB memory. During the runtime of the application, a single Kinect motion capture device was used to capture the user's performance. Finally, during the application runtime, the system ran at 25 frames per second, with latency less than 160 ms.

During the capturing process, different motion styles are recorded, where each style consists of both locomotion and non-locomotion sequences. The different styles consist of angry, happy, sad, tired, and sneaky. In addition, each motion style is captured for walking, jumping, conversing, golf swing, reaching, kicking, punching, and hand waving. The variation in each motion style is based on the ability to generalise the correlation process between the natural and stylistic representations of each motion style. In addition, a closely related normal representation of each motion is captured for use as a reference motion for building the correlation. Table 4.1 lists the number of frames captured for each style. Because the stylistic and normal motion sequences are aligned and synchronised, the final number of frames are equal at every pair of corresponding motion sequences.

Styles	Angry	Happy	Sad	Tired	Sneaky
Frames	3848	3534	4776	3973	4059

Table 4.1: The amount of data in frames of each captured stylistic motion.

During the runtime of the application, the user is able to perform each natural motion.

Based on the specified stylistic behaviour, the system synthesises the stylistic perspectives for the input motion. Such a solution could potentially work with various users. However, even if all of the predefined motion sequences contained in the database are retrieved from the same performer, a calibration process between the input skeletal data and the skeletal data contained in the database should be defined. The calibration process is responsible for mapping the input positions and orientations of the extracted skeleton of the user to the defined skeleton. The skeletal data retrieved from the CMU motion capture database Carnegie Mellon University (2014) is represented in the ASF format. Each skeleton example records the lengths of individual bones. The skeleton of the virtual character contains 24 bones (head, thorax, upper neck, lower neck, upper back, lower back, left and right clavicle, humerus, radius, wrist, hand, hip, femur, tibia, and metatarsal).

For the calibration process, the system first measures the world position of the user's skeletal joints using a single "T" pose. Based on this information, the 3D positions of the inboard joints relative to the world coordinate frame in the "T" pose are mapped via forward kinematics. The locations of the skeletal joints in the coordinate system of the inboard joint are then found by computing the difference between the location of the inboard joint and the skeleton's joint data to the world coordinate frame. Finally, the motion capture database is processed by computing the three-dimensional location of the input data c_t retrieved from the motion capture device, which correspond to the motion capture data for each frame in the database q_t according to:

$$c_t = f(q_t; \tilde{g}, \tilde{z}, h_0) \tag{4.14}$$

where $f(\cdot)$ is the forward kinematics function that is responsible for computing the position from the joint angles of the current frame q_t given the user's skeleton model $\tilde{g}$ and $\tilde{z}$ is the location of the skeletal data retrieved from the motion capture device, which is relative to the inboard joint. In addition, h_0 denotes a default root position and orientation. Finally, based on the calibration process, the system can synthesise the desired stylistic perspective of an input motion when users with different morphological variations are used for the capturing process. Figure 4.6 shows the synthesis of the stylistic motion while different users perform the desired motions.

4.5.2 Experimentations

Since it is required a generalisation of the presented methodology the following experiments were conducted. Firstly, it is examined the ability of synthesising a regular motion while the user performs the stylistic motions (see Figure 4.7). This inverse process was chosen

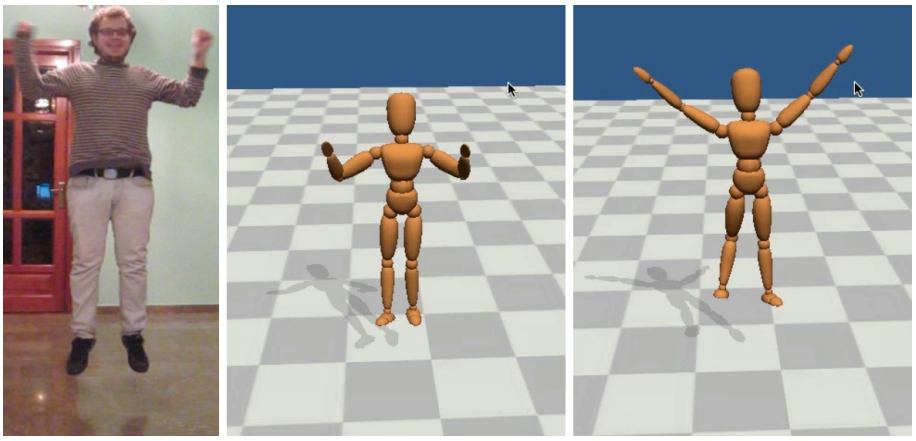

Figure 4.6: Stylistic postures of the character generated from a different user.

for showing the efficiency of the presented methodology to provide the desired results while a style motion is used as input. Moreover, this simple experiment shows the flexibility of the system as well as the accuracy of the motion correlation process to provide the expected results.

In this case, the RBF which is responsible for computing the style components trained by using the inverse process. It should be mentioned that the inverse process works well since the system is able to synthesise the desired regular motion while different styles were performed by the user. Additionally, a second experiment that conducted was to synthesise a stylistic motion while the user performs a different stylistic motion. In this case, considering an input stylistic motion e.g. angry walk, and the synthesised stylistic motion e.g. a happy walk, the system trained by those two motion styles, synthesising the motion that contains the desired stylistic behaviour. Finally, since the training of the RBS is always computed based only on two reference styles, another experiment was the ability of synthesising a motion sequences while the user performs a different stylistic behaviour than those used for the training process. For example, training the RBF with two motion styles such as an angry style, and a sad style, while the user performs a different style such as happy style, the system is able to synthesise a new motion style variation that is not included in the reference style variations.

Based on the previously mentioned experiments the following can be stated. Firstly, while it is desired the inverse style motion synthesis, the system is able to synthesise the desired motion. Secondly, while the system is trained by two different styles, it is able to synthesise the desired stylistic motion while the user performs the trained input stylistic motion. It should be noted that these results are based on visual assessment. Although, additional cross validation techniques, such as computing the reconstruction error can

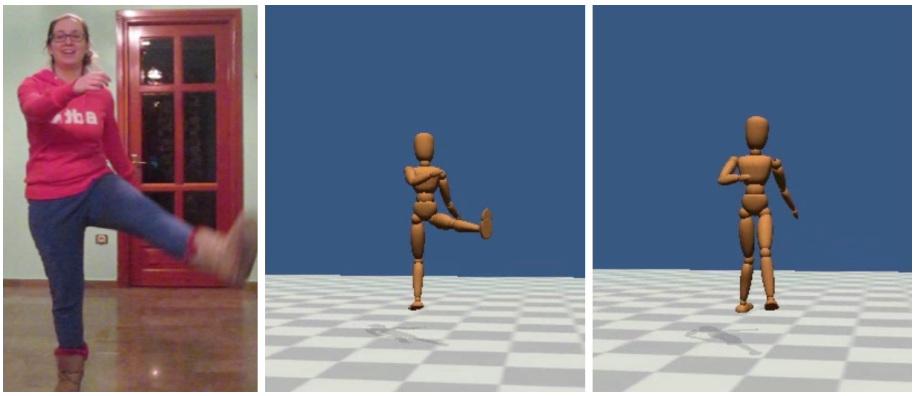

Figure 4.7: The user performs a motion style (left). The system captures the motion style (middle), and synthesise a regular motion (right).

also be used for retrieving quantitive results. Finally, the most interesting aspect of the presented methodology is the ability of synthesising a new motion sequence while the user performs a different motion style than those two used for the training process. Therefore, in this case it should be mentioned the following. While the system is trained by two motion styles, and the user performs a different style than those used for the training process, the resulted motion keeps the stylistic variation that is required. Although, as it was observed, depending on the training motion styles, as well as on the user's performance, the new synthesised motion can be characterise either as sharper or as blunt.

This is explained as follows. While the system is trained by two style behaviours the characteristics of the target style are computed according to the characteristics of the reference style. Hence, while the user performs a new motion that can be characterised as sharper in movements i.e., a motion where the end-effectors move faster and abruptly, compared with the reference motion, it was observed that the new synthesised stylistic motion is sharper than the target stylistic motion. Similarly, while the user performs a motion style that can be characterised as blunt (i.e., a motion where the end-effectors move slower and less abruptly, compared with the reference motion, the new synthesised motion), is observed being more blunt. Hence, based on these experiments it is possible to conclude that the synthesised motion styles does not only depend on the reference motion styles used during the motion correlation process, but it is also dependent to the user's performance.

4.6 Discussion

In this chapter, a methodology for synthesising real-time style variation from input motions was presented. The presented method uses a motion style transfer process that is designed to work with real-time computation. A parameterised MAP estimation was implemented, which is responsible for synthesising the desired motion of the performer based on the stylistic parameters. Motions can be captured in real time for the desired stylistic sequence while the user performs a normal motion. This can be quite beneficial in the film industry where it is difficult to manage a large number of actors performing different stylistic motions due to budget and production constraints. Moreover, assuming that the stylistic content is captured in real time, it is possible to directly identify incorrectly captured sequences.

A limitation of this method is its inability to generalise the motion correlation process because every human performs the same action with a different motion style. Therefore, every user requires a representation of the normal motion sequences, which must be closely related to the existing stylistic motions contained in the database. In cases where simple stylistic behaviours are desired, such as designing a sneaky walking motion by capturing natural walking, it is possible to achieve the desired results, as shown for different users in the real-time stylistic motion synthesis process.

Chapter 5

Fingers Motion Synthesis

With the motion synthesis methodologies presented in the previous two chapters it is possible to synthesise in a natural looking way the motion of a virtual character. However, the synthesised motion sequences do not include details, such as the motion of the face and the motion of the fingers. Thus, while capturing the whole body of a performer, both the fingers and the face of the character are represented with static poses. The required realism of motion sequences can be improved by assigning details to the virtual character's body, such as the motion of the character's fingers Jörg et al. (2010) Kendon (2004) Samadani et al. (2011b). Therefore, two methodologies for estimating the motion of a character's fingers are presented in this chapter. The first method focuses on the ability to estimate the motion of the character's fingers based for off-line computation, while the second methodology achieves estimation and synthesis of the motion of the character's fingers in real-time.

5.1 Introduction

Creating finger motions for a virtual character is a complex and time-consuming process because the human hand is highly articulated and contains many degrees of freedom (DOF). However, in recent years, many solutions for capturing human hand motions and then retargeting this motion to any virtual character have been produced. As various perceptual studies have shown in the past several years, hand details enhance the realism of the generated motion Jörg et al. (2010) Kendon (2004) Samadani et al. (2011b). A simple example showing a character both with and without the finger posture is illustrated in Figure 5.1. Thus, as it can easily be illustrated that key meaning may be lost when finger motion is omitted.

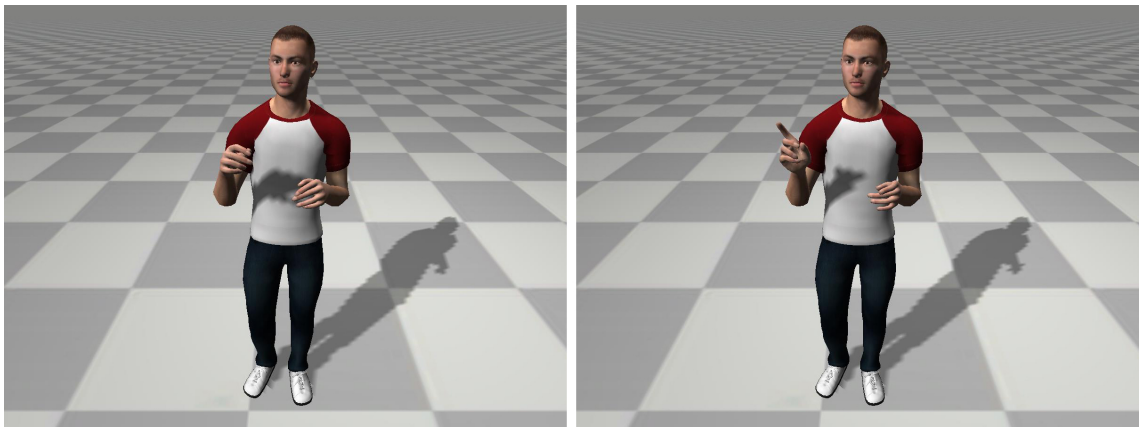

Figure 5.1: A character that does not include the finger postures changes the meaning of the pose. A character without finger postures (left), and the same character with finger postures (right).

Gesturing and conversational characters can be found in various applications of virtual reality, such as in telepresence, video games, and films Kalra et al. (1998). However, because obtaining the desired result when designing a character's hand motion is time-consuming or expensive, new techniques that allow the automatic assembly of finger motions to a virtual character are necessary. Hence, methodologies that generate a character's hand motion have been proposed in recent years. Although, as examined in on-line motion reconstruction, and more specifically in computer puppetry techniques Sturman (1998) Shin et al. (2001), methods that estimate the motion of the character by using as minimal information as possible are always desirable.

In the presented solutions, it is first introduced a different way to segment the motion data that is responsible for splitting the motion sequences into gesture and non-gesture phases. In addition to the motion segmentation process, a distance metric that determines whether the input segment consists of a gesture or on a non-gesture phase is introduced. Based on the ability to classify the input motions as gestures or not gestures, the system is responsible for searching directly within the gestures or the non-gesture collection of motion data to retrieve the most appropriate motion. However, in having those two different collections of data, the most appropriate method to estimate the required motion is required. In this case, the system searches to find the most valid gesture type in cases where the input segment belongs to a gesture phase. Thus, a novel distance metric that computes the motion features of the input motion with those produced by analyzing the existing motion data is introduced. Having, found the gesture type, the system now is able to search through the collection of the recognised gesture type to which the gesture belongs, to find the most valid motion. This is achieved by using a simplified distance

metric. Although the input segment does not consist of the collection of key gestures, the same distance metric is used for retrieval of the character's most valid non-gesture motion. A simple example of finger gestures that have been synthesised with the presented methodology is illustrated in Figure 5.2.

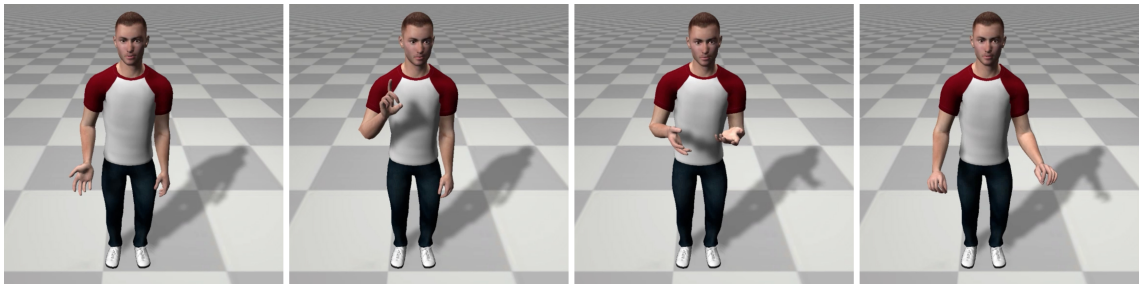

Figure 5.2: Examples of finger gestures synthesised with the presented solution.

An extension of the aforementioned methodology for computing the motion of the character's fingers in real-time is presented. The presented methodology relies on the analysis of small segments of existing motion data that contain the required motion of the character's fingers, and by computing global features that characterise each segment. Having analysed those segments, the motion synthesis process relies upon the ability to compute a simple distance function based on the computed features of human motion.

In this case two basic limitations present themselves. Firstly, since the motion synthesis process is implemented to work with the Microsoft's Kinect motion capture device, it requires a set of motion features that will be used for the motion estimation process. It should be noted that the Kinect motion capture device does not compute the orientation of the performer's wrist, making the motion estimation process quite difficult. The orientation parameter can be quite beneficial in the gesture recognition process, as indicated in various studies Yoon et al. (2001) Liu et al. (2004) Elmezain et al. (2008) Jörg et al. (2012a). Therefore, additional motion features are examined for estimating a valid motion of the character's fingers. Secondly, since the motion of the character's fingers is split in to small segments, the required meaning of a long motion sequence may not be clearly understood. For that reason, a highly conditioned cost function is implemented. Thus, the system is able to synthesise long motion sequences that retain the required meaning. A simple example of the real-time motion synthesis process is shown in Figure 5.3.

To the best of our knowledge, the finger motion synthesis process has not yet been solved in such a way. In addition, while the datasets that contain different finger gestures will increase in the future, estimating the motion of the character's fingers based on the global feature analysis techniques could be quite useful as explained above. Moreover,

Figure 5.3: The performer (left) is captured by the system, and the motion is mapped to a virtual character (middle). In the presented methodology, the system computes the motion of the character's finger (right), which is displayed simultaneously with the character's body.

in cases that require real-time implementation, such as in estimating the finger motion during the motion capture process, the analysis of the motion data and the use of global features to describe each gesture of the character can also be more efficient. Finally, it should be mentioned that the main advantage of the generalization process of existing motion data is that it makes it possible to minimize the error in estimating the correct motion comparing with a previous solution Jörg et al. (2012a).

5.2 Off-Line Motion Synthesis

Various steps are involved in synthesising the most appropriate motions. More specifically, in a pre-processing step, given the motion sequences that contain the required motions, it is first necessary to use an efficient segmentation method, e.g., by separating the gesture from the non-gesture phases of the character's hand motions (Section 5.2.1). Then, when the system has the motion segments, it must be able to search for the most appropriate features (Section 5.2.2) of the motion. These include estimating whether the input motion belongs to a gesture or to a non-gesture phase (Section 5.2.3). In this case, a feature-based distance metric is used for this searching process. Having found the collection of motions to which the input gesture belongs, the system must estimate the most probable gesture type of the input motion (Section 5.2.3). In this case, when the input segment belongs to a gesture motion, the system can estimate the gesture type by using an additional distance metric based on the analysis of motion features. Having found the gesture type or that the input segment consists of a non-gesture motion, the system extracts the most valid motion of the character's finger using a simpler distance metric (Section 5.2.3) than that proposed by Jörg et al. Jörg et al. (2012a), for estimating the most relevant motion. Having found the most valid motion for each input segment, the system synthesises the new motion of

the character's fingers. This is achieved by assigning the final motion synthesis process to the standard motion graphs pipeline, which is responsible for synthesising the required transitions, as proposed by Kovar et al. Kovar et al. (2002a). Figure 5.4 illustrates the pipeline of the presented methodology.

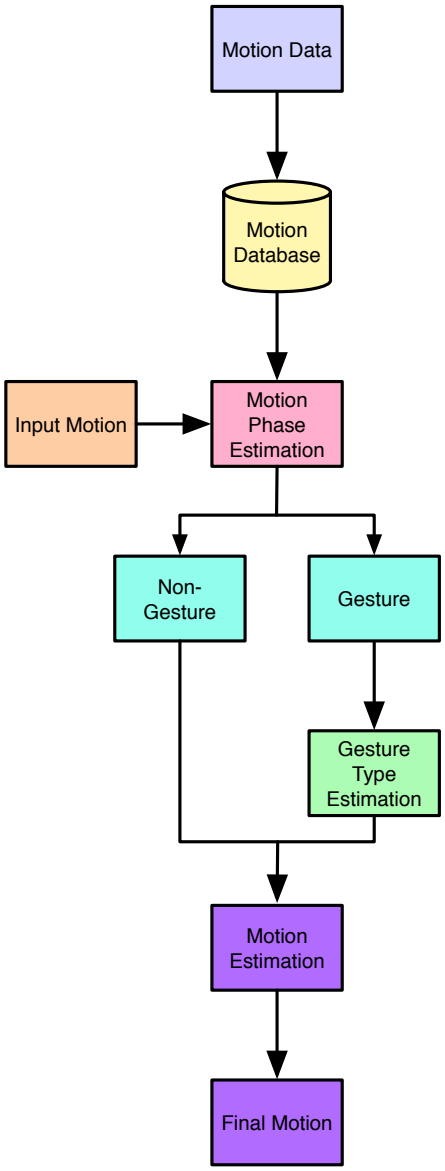

Figure 5.4: The pipeline of the presented methodology.

In this case, the following should be noted. While the gesture estimation process is separated into sub-problems, the computational cost is minimised. More specifically, the search process is simplified. The present methodology of searching for an input gesture segment individually by using the reference data contained in the database, as proposed by Jörg et al. Jörg et al. (2012a) i.e., searching for the motion valid segment by comparing the input segment with each of the 64 motions (8 gesture types $\times$ 8 motions) is not

used. Instead, in the presented methodology, the system compares an input segment against two computations of motion features and one computation against the motion data for the gesture phases. More specifically, the presented methodology first performs two comparisons of the gesture and non-gesture phases, eight comparisons of the gesture types, and eight comparisons of the motion segments. Therefore, the number of comparisons in the search process is reduced to 18, which makes the presented approach efficient in comparison to the previous solution.

5.2.1 Analysing and Segmenting Finger Gestures

In this case, it is required to separate each motion sequence into three phases: from the preparation (the arm moves away from its rest position) to the pre-stroke hold, from the pre-stroke hold to the post-stroke hold (the arm remains static), and from the post-stroke hold to the retraction (the arm moves back to its rest position). This is achieved by segmenting the motion data based on the maximum velocity of the wrist, instead of the minimum velocity of the wrist, as used in Jörg et al. (2012a) and Levine et al. (2009). It should be noted that by segmenting the motion based on the maximum velocity, as it is shown in the next section (seel also Figure 5.5) it is possible to obtain the motion where the fingers start to move. Thus, it can be said that the time period between the maximum velocities is the one that provides the start and the final point of a finger gesture. To conclude in those phases the next steps evolved as presented in the following subsections.

Motion Analysis

Having synchronised the motion sequences to estimate the time period in which a gesture begins to appear, by using the dynamic time warping function presented in Section 4.3.1, it is necessary to compute the necessary features of motion that can provide this ability. In this case, the calculation of the kinetic energy that the finger motions produce is a possible solution. However, by assuming that the mass of the character's hand is a constant variable, it was considered to compute the angular velocity of each of the degrees of freedom of the character's fingers, according to:

$$V(t) = \frac{1}{N \times D} \sum_{n=1}^{N} \sum_{d=1}^{D} v_{n,d}(t) \tag{5.1}$$

where $v_{n,d}$ is the angular velocity of the $d-th$ degree of freedom of the character's hand skeletal model for $d = 1, ..., D$, where in the 3D model that used $D = 25$. Moreover, n denotes the $n-th$ motion sequence that is used for the gesture type analysis process,

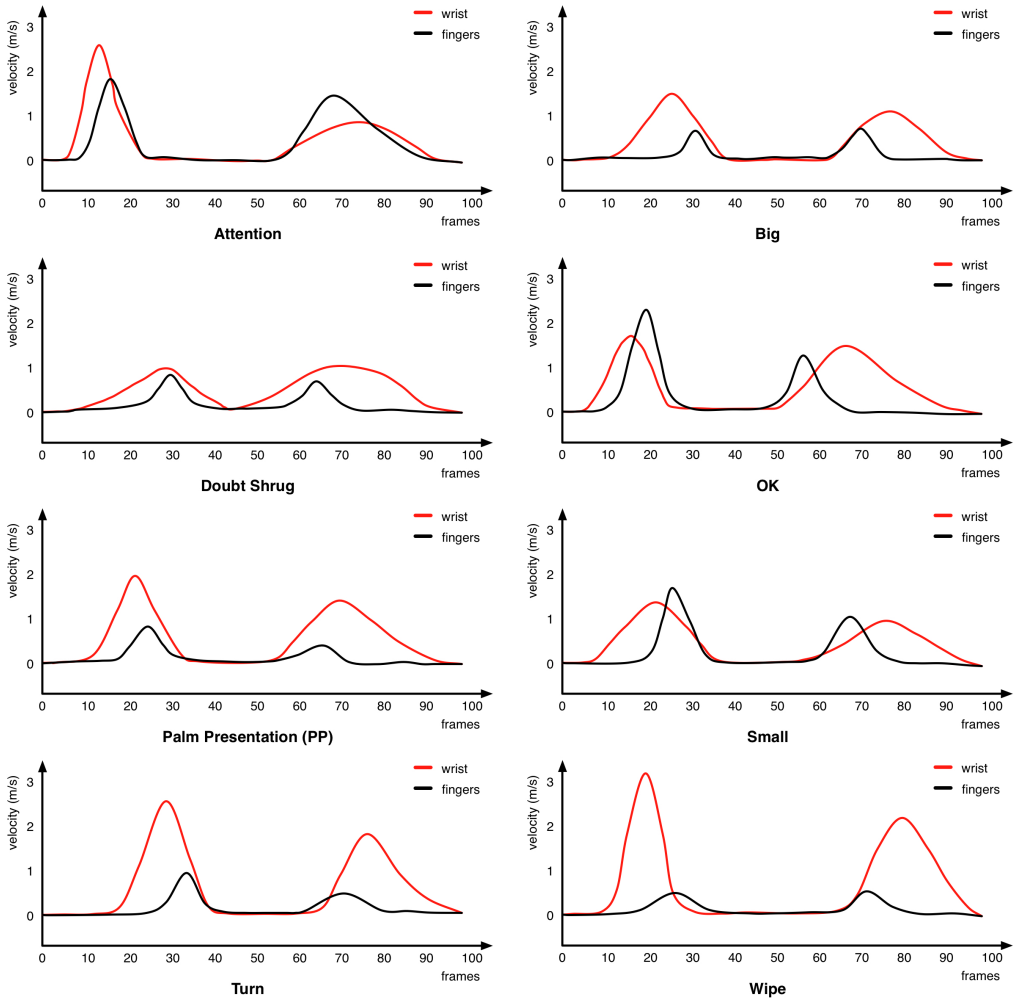

Figure 5.5: The generalised representation of the angular velocity parameter of the character's fingers and wrist for each of the different gesture types that was examined.

where in our case $n = 8$, which is the actual number of motion sequences that belongs at each gesture type, and $t = 1, ..., T$ denotes the total number of frames of each motion sequence. Finally, $V(t)$ is the angular velocity calculated for each frame for each degree of freedom of each motion that belongs to a gesture type of the character's hand. Therefore, for each of the motion sequences of a gesture type, the generalised representation of the gesture's motion is determined.

In the same manner as used previously to analyse the motion of the character's wrist, the velocity of each motion of a specified gesture is calculated. Then, for every motion of the character that belongs to a specified gesture, the mean value of the velocity at each time step for retrieving the generalised representation of the motion data is calculated. The resulting graphs that illustrate the representation of the angular velocity parameter for the motion of the character's fingers and the velocity of the character's wrist are shown in Figure 5.5.

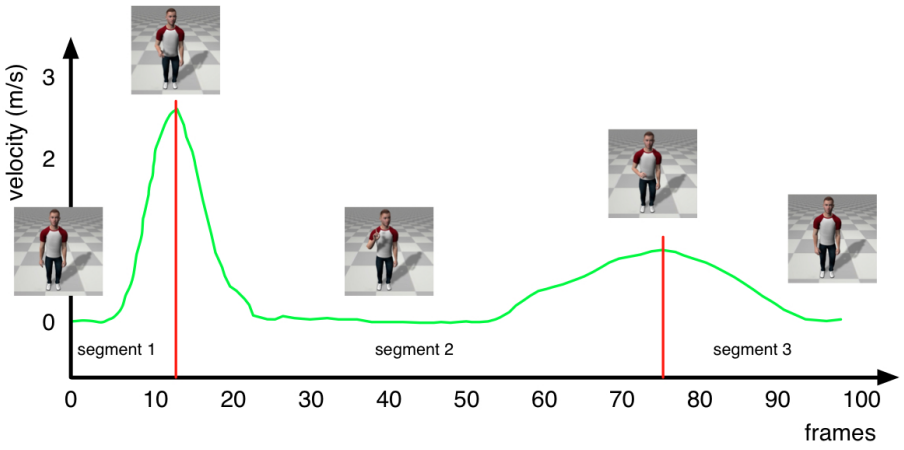

Figure 5.6: An example of a graph that plots the velocity of the character's wrist. The phases are split according to the proposed method. Phases 1 and 3 are the preparation and the retraction respectively, whereas phase 2 is the meaningful part of the gesture.

Motion Segmentation

The process of correlation of the motion of the finger and the motion of the wrist is important. Since the input motion is segmented according to the wrist's parameters, this correlation is necessary for the system to efficiently synthesise the new motion, while keeping the necessary temporal parameters. Therefore, undesired effects such as incorrect synchronisation should be eliminated.

In this case, in examining the graphs illustrated in Figure 5.5, one can assume that the most valid way to segment the motion of the character's fingers is while the character's wrist crosses its maximum velocity. This is because the character's fingers begin to react in order to perform a gesture during the period in which the wrist is located between the pre-stroke hold and the stroke phase of a gesture. This in-between phase (phase 2, see Figure 5.6) can be characterised by where the meaning of the gesture appears in the first frame of the phase, and disappears in the last frame of the phase. Therefore, the required meaningful phase of a motion segment is split efficiently by using the proposed segmentation method that relies on the analysis of the motion capture data.

In this case, it should be noted that to automatically segment the motion data into the desired phases, as inspired by Fod et al. Fod et al. (2002) creates segment boundaries when a shift from fast motion to slow motion is detected. Therefore, for each frame of motion data, it is computed $z = \|v_{wrist}\|^2$, where v_{wrist} denotes the square of the velocity of the character's wrist.

Therefore, by using the dataset provided by Jörg et al. Jörg et al. (2012a) that contains 64 motions (eight repetitions $\times$ eight gesture types) that can be found in Jörg (2014), a

total number of 197 segments is generated. More specifically, 59 of the 64 motions are separated into three phases, while the remaining motions (five motions) are separated into 4 phases, since the velocity slope is appeared twice. It should be mentioned that the lengths of the segments does not influence negatively the motion synthesis process, as in Jörg et al. (2012a). The reason is that the motions with the short post-stroke hold phases can be separated efficiently while the segmentation evolves during the maximum values of the wrist's velocity. Moreover, for the motions that were separated into four phases, each segment that is between the maximum wrist velocities is considered to be a gesture phase. Finally, it should be mentioned that, with this segmentation process, it is possible to effectively estimate whether the input segment consists of a gesture phase or a non-gesture phase.

Evaluating Motion Segmentation

A motion segment should be efficiently synthesisable. This means that each pair of corresponding motions should be able to transit smoothly during the motion synthesis process. Thus, a segmentation process should be able to provide the ability to split the motion data into parts that minimize the transition cost.

For that reason, the transition cost of every pair of segmented data based on the proposed methodology, as well as on the methodology used, in Jörg et al. (2012a) was computed. This is the transition cost of each pair of motion sequences, such as from every segment that belongs to phase 1 with any segment that belongs to phase 2, and from every segment that belongs to phase 2 with any segment that belongs to phase 3. Additionally, since each of the meaningful phases (phase 2) will appear after any other segment that belongs to the same phase, an additional calculation is made to compute the transition cost. Thus, it is computed the transition cost between the last frame, t_{last}, of an origin segment A and the first frame, t_1, of a target segment B by the following equation:

$$c_T = \frac{1}{D}\sum_{d=1}^{D}(A_d(t_{last}) - B_d(t_1))^2 \tag{5.2}$$

where $d = 1, ..., D$ is the total number of degrees of freedom of the character's hand skeleton. Having computed the transition cost function for each pair of phases, the mean average of each pair of phases is then calculated. The results of this evaluation process are summarised in Figure 5.7. The following should be noted when examining these results. Firstly, the proposed segmentation method minimizes the transition cost between two motion segments comparing to Jörg et al. (2012a), where on average the transition

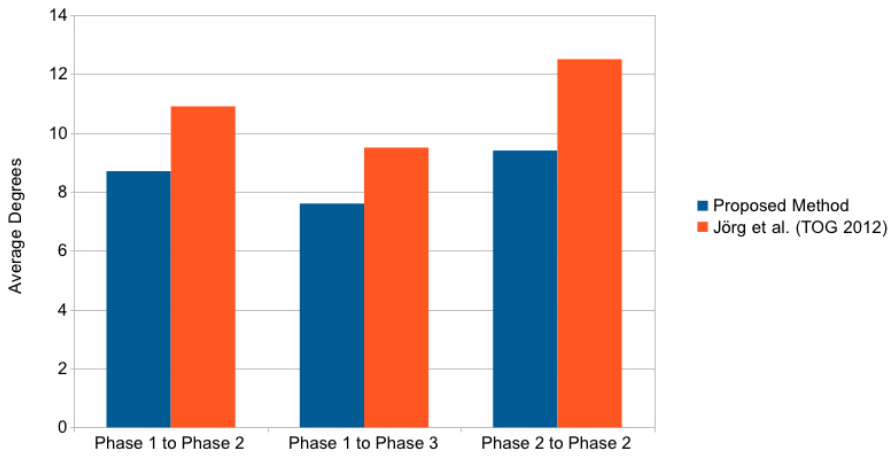

Figure 5.7: The results obtained from the transition cost evaluation between the proposed segmentation method, and that used in Jörg et al. (2012a).

cost has been reduced by 25% while the transition process evolves both from phase 1 to phase 2 and from phase 1 to phase 3. Moreover, the interesting results arise when evaluating the gesture-to-gesture transition, such as while a motion segment from phase 2 is required to transit to a motion segment that belongs to phase 2. In this case, the proposed segmentation methodology achieves a much better transition since the transition cost function is minimised by average of 33% Jörg et al. (2012a). Therefore, it is assumed that the proposed method ensures a smoother transition that will make the motion synthesis process more natural looking. It should be noted that the average transition cost is approximated by computing the c_T both while using the presented segmentation and the segmentation method used in Jörg et al. (2012a).

5.2.2 Computing Motion Features

Various features provided by the character's wrist can be computed. More specifically, for each input segment s_i, it was considered the distance (the travelled distance) d_i, the average orientation $\bar{o}_i$, the average velocity $\bar{v}_i$, the average angular velocity $\bar{av}_i$, the average acceleration $\bar{a}_i$, and the average angular acceleration $\bar{aa}_i$. It should be mentioned that a variety of other features could be used, such as the high, the temporal and the spatial extent, and also the curvature. The aforementioned motion features were computed as:

$$d_i = \frac{1}{n}\sum_{i=1}^{n} \|p_i - p_1\| \tag{5.3}$$

$$\bar{o}_i = \frac{1}{n}\sum_{i=1}^{n} \|o_i\| \tag{5.4}$$

$$\bar{v}_i = \frac{1}{n}\sum_{i=1}^{n} \|v_i\| \tag{5.5}$$

$$\bar{av}_i = \frac{1}{n}\sum_{i=1}^{n} \|av_i\| \tag{5.6}$$

$$\bar{a}_i = \frac{1}{n}\sum_{i=1}^{n} \|a_i\| \tag{5.7}$$

$$\bar{aa}_i = \frac{1}{n}\sum_{i=1}^{n} \|aa_i\| \tag{5.8}$$

where p_i, o_i, v_i, av_i, a_i, and aa_i denote the position, orientation, velocity, angular velocity, acceleration, and angular acceleration of the character's wrist at the $i-th$ frame of the motion segment. Having computed these features each of the motion segments s_i is represented as $s_i = \{d_i, \bar{o}_i, \bar{v}_i, \bar{av}_i, \bar{a}_i, \bar{aa}_i\}$. Having collected these features, one must find those features that are necessary to distinguish a gesture from a non-gesture phase of the input motion segment, as well as those features that can be used to estimate the most valid gesture of the character. Both of these methods are presented in the following subsections.

5.2.3 Finger Motion Estimation

This subsection describes the finger motion estimation process. More specifically, it is first presented the separation process of the motion segments into gesture and non-gesture phases. Then, it is shown the searching process through the gesture phases to find the most appropriate gesture type. Finally, it is presented a simplified distance metric for estimating the most probable motion.

Gesture Phase Estimation

The computation that shows whether the input segment consists of a gesture phase is one of the most important parts of the presented solution because an incorrect estimate directly affects the selection of the most appropriate motion. Therefore, careful consideration is required. As mentioned earlier, the majority of the motion sequence is segmented into three phases, with two of those phases marked as non-gesture phases, and the middle phase marked as a gesture phase. In this case, by plotting various features against each standard deviation, as well as by evaluating each gesture contained in the database, it is possible to separate the gesture with the non-gesture phases, either by using the wrist's orientation or the wrist's position. More specifically, motion segments that belong to attention, big, small, turn, and wipe, can be recognised by the position of the character's wrist. Motion

segments that belong to the doubt shrug, and ok gesture can be recognised by the wrist's average orientation. The plots that distinguish the gesture from the non-gesture phases that are used for this process are illustrated in Figure 5.8.

The process that distinguishes the gesture and the non-gesture segments can be easily computed. In this case, given an input segment s_i for distinguishing the gesture from the non-gesture phases, the following condition is imposed:

$$s_i = \begin{cases} gesture & \text{if } g_g^d \leq g_{ng}^d \;||\; g_g^r \leq g_{ng}^r \\ non-gesture & \text{otherwise} \end{cases} \tag{5.9}$$

where g_g^d, and g_{ng}^d represent distance between the distance features of the gesture and non-gesture phases respectively, and g_g^r, and g_{ng}^r represent the distance between the orientation feature of the gesture and the non-gesture phases respectively. Thus, it is required to be computed the distance of the examined features g_D, and g_O by use of the following equations:

$$g_D = \frac{(\bar{d}_i - \bar{D})}{\sigma_D^2} \tag{5.10}$$

$$g_O = \frac{(\bar{o}_i - \bar{O})}{\sigma_O^2} \tag{5.11}$$

where $\bar{d}_i$ represents the distance of the wrist and $\bar{o}_i$ represents the average orientation of the wrist, both of those being computed from the input motion segment. Moreover, $\bar{D}$ represents the mean value of the distance, and $\bar{O}$ represents the mean value of the orientation, where both of those are provided by the motion capture data contained in the database. Additionally, σ_O, and σ_D represent the standard deviations computed from the registered reference orientations and distance respectively provided by the character's wrist. Finally, it should be noted that both g_D and g_O are computed for both the gesture and non-gesture phases of $\bar{D}$ and $\bar{O}$.

Gesture Type Estimation

While knowing in advance that the input segmented motion belongs to a gesture phase, it is now required a method to estimate the gesture type. In this case, it was not considered directly estimating the most probable motion because the presented methodology addresses the need to analyse the global features that are provided by analyzing the existing motion data. Therefore, it was assumed that it would be helpful to first recognize the type of gesture and then to estimate the most probable motion. Hence, not only can the system estimate the most valid gesture type, but also it can retrieve the motion segments that best match the input motion.

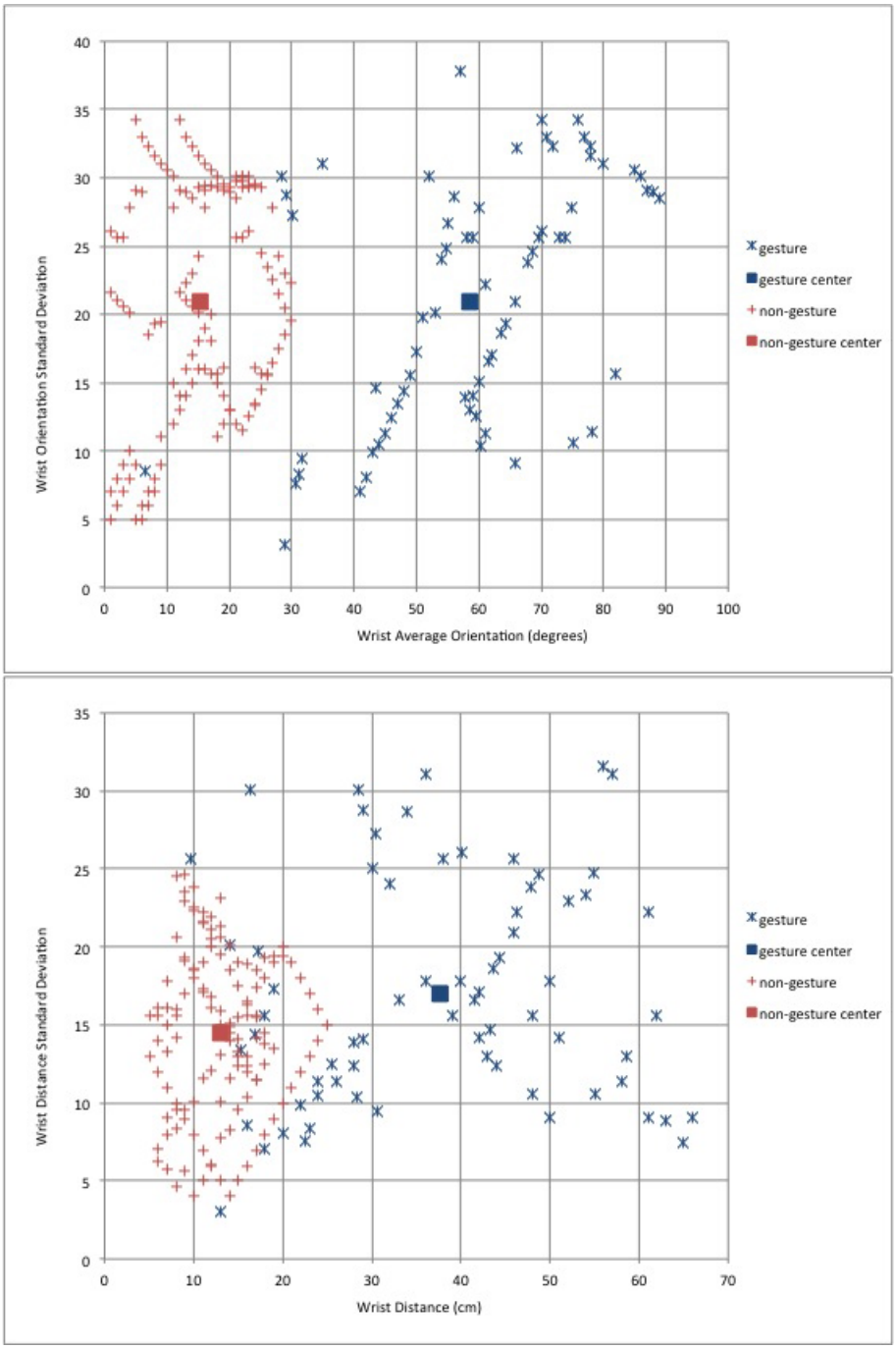

Figure 5.8: The resulting plot based on the character's wrist average orientation (upper) and distance (lower) against its standard deviations.

Next, based on the features that can be computed from the character's wrist, the system estimates the most probable gesture type based on the distance between the features that are produced from the input motion and the mean value of each reference gesture type's features. In this case, each feature of the reference motion data is plotted (Figure 5.9), such as being able to understand how each feature of each gesture type is distributed. As can be concluded from those plots, there is a possible distinction between each feature of each gesture type, although there are intersection areas that depend on the features. In this case, it was assumed that, if the largest number of motion features is used, the most accurate estimation of the gesture type can be computed. This assumption is confirmed in an evaluation process. Therefore, to estimate the most appropriate gesture type, a distance function that accounts for each of the examined features was designed.

More specifically, each input gesture segment s_i is assigned all of those features that are mentioned in Section 5.2.4. Therefore, the input segmented motion is represented as $s_i = \{d_i, \bar{o}_i, \bar{v}_i, \bar{av}_i, \bar{a}_i, \bar{aa}_i\}$. Thus, estimating the most probable gesture type g_T is computed as follows:

$$\begin{aligned} g_T = & \frac{(d_i - \bar{D})^2}{\sigma_D^2} + \frac{(\bar{o}_i - \bar{O})^2}{\sigma_O^2} + \frac{(\bar{v}_i - \bar{V})^2}{\sigma_V^2} + \frac{(\bar{av}_i - \bar{AV})^2}{\sigma_{AV}^2} \\ & + \frac{(\bar{a}_i - \bar{A})^2}{\sigma_A^2} + \frac{(\bar{aa}_i - \bar{AA})^2}{\sigma_{AA}^2} \end{aligned} \tag{5.12}$$

where $\bar{D}$, $\bar{O}$, $\bar{V}$, $\bar{AV}$, $\bar{A}$, and ($\bar{AA}$ denote the mean values of distance, orientation, velocity, angular velocity, acceleration, and angular acceleration, respectively, which are provided by the reference motions. Moreover, σ_D^2, σ_O^2, σ_V^2, σ_{AV}^2, σ_A^2, and σ_{AA}^2 represent the standard deviations computed from the reference registered motion features provided by the character's wrist distance, orientation, velocity, angular velocity, acceleration, and angular acceleration, respectively. Therefore, based on Equation 5.12, the most probable gesture type is that for which the value of g_T is at a minimum. Finally, it should be mentioned that such a method can be solved by assigning the most appropriate values to a decision tree. However, it was chosen to compute the gesture type of the input motion based on a more generalised method, as it is provided by the distance metric, instead of using bounding limits for each motion feature, resulting in a highly constrained searching process.

Motion Estimation

Having found that the input motion belongs to either a gesture or a non-gesture phase, a method is required to estimate the motion that best matches the input motion. This is achieved by assigning the exact motion estimation process to a distance metric. After the

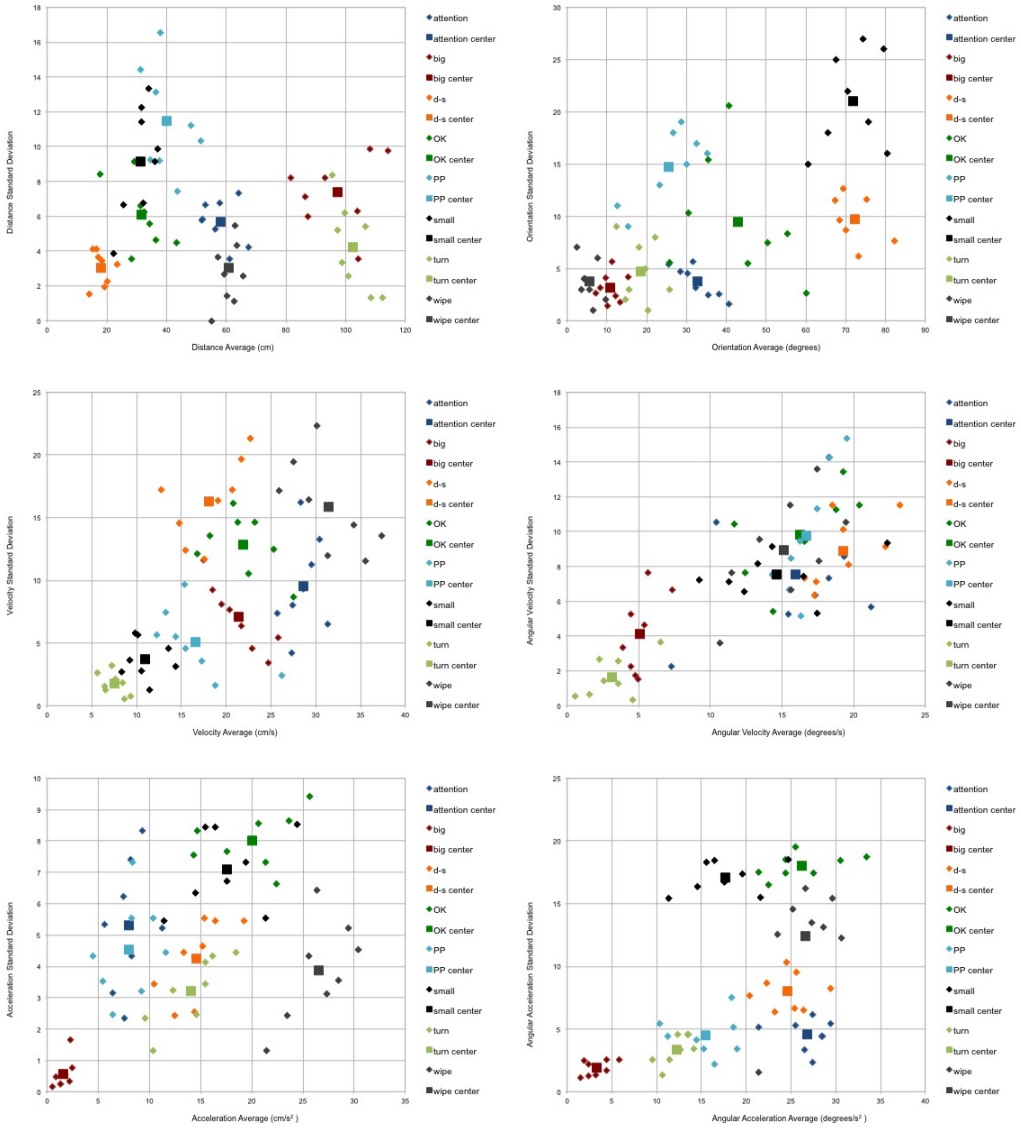

Figure 5.9: The resulting plot based on the character's wrist average orientation (upper) and distance (lower) against its standard deviations.

most probable gesture type is estimated, it is not necessary to search again through the motion collection of the estimated gestures. This is especially true for the gesture phases. Therefore, by assigning any one of those gestures to the input motion, it is marked as a correct decision. However, the most probable motion that is expected to fulfil at least the closest spatial parameters of the input motion must be estimated. Thus, the ability to assign the final decision of the motion estimation to an additional search step is very helpful because it not only ensures the meaning of the synthesised motion, but also ensures the naturalness that is required of the synthesised motion.

For that reason, it is presented a simplified distance metric (compared to that of Jörg et al. Jörg et al. (2012a)) because it does not account for any additional weights on the input parameters. More specifically, it must have a method to estimate the reference motion segment S_D that is contained in a database and that best matches the input motion segment S_I. Therefore, the following equation is used:

$$D_M = \frac{1}{N}\sum_{t=1}^{N}(R_I(t) - R_D(t))^2 \tag{5.13}$$

where $R_I(t)$ and $R_D(t)$ denote the orientation of the input motion and each of the motions that are contained in the database at the $t-th$ frame for $t = 1, ..., N$. For the computational process of the distance metric, a dynamic time warping function is used, such as each of the R_D reference motions, in order to have the same length as the input motion R_I. Moreover, this distance metric is used to search for the non-gesture segments. Finally, it should be noted that no additional computation was implemented in the presented solution, such as a transition cost function. This is since, it was assumed that the motion that best match to the reference one, is the most valid motion that should be synthesised.

5.2.4 Evaluations and Results

In this section, it is presented the results of the evaluation performed during each step of the motion estimation process as it was presented in the previous subsections. It is evaluated almost every step of the presented motion estimation process. Starting from the ability of recognizing whether the input segment belongs in a gesture or a non-gesture phase, as presented in Section 5.2.3, it is evaluated the accuracy of the presented methodology by using the methodology that is described.

Evaluating the Motion Phase Estimation

Given an input segment in a first step, the system is responsible for estimating whether the input segment consists of a gesture or a non-gesture phase of motion. Therefore, by using the average orientation and distance of the character's wrist, it is possible to estimate the phase in which the gesture belongs. As mentioned earlier, an incorrect computation of the motion phase directly affects the motion estimation process. For example, consider an input segment that belongs to a gesture phase and assume that the system recognizes this gesture as a non-gesture phase. In that case, the system will search directly for a non-gesture motion to assemble for the character's hand.

In this case, to evaluate the accuracy of the motion phase estimation, the leave-one-out cross validation methodology was conducted. More specifically, by leaving out one of the segments (a segment that consists of either a gesture or a non-gesture phase), it was computed if the system identifies correctly the input motion segment. The same procedure was repeated for each of the segments (197 segments) that are contained in the database. In addition, it should be noted that for every iteration, the new values that describe the standard deviations of the motion features used were calculated. The results obtained from this evaluation process are presented in Figure 5.10. They are quite promising. The presented method can estimate correctly the gesture phase with accuracy of approximately 98%.

To evaluate how accurate such a method can be, very similar results were obtained when evaluating the motion phase estimation and when using the 8 gesture types for the training process with motion segments from the other datasets (directions, debates, and conversations databases) that can be found in Jörg (2014). While the gesture database is used for the training process, the estimation of the correct gestures in the gesture dataset is approximately 93% in the directions dataset, 96% in the debate dataset and 95% in the conversation dataset. This results approximated by assigning each segment to the searching function (see Equation 5.9), which is responsible for distinguishing the gesture with the non-gesture phases. After comparing the difference between the gestures dataset and any of the other datasets, it can be stated that these differences result from the inability of the existing motion data, which is used for the distinction between the gesture and non-gesture phases, to describe the large variety of motions that can be produced by the character. However, because the results are quite high, it is clear that the method works well. It should be noted that a greater amount of data may increase the gesture phase estimation process.

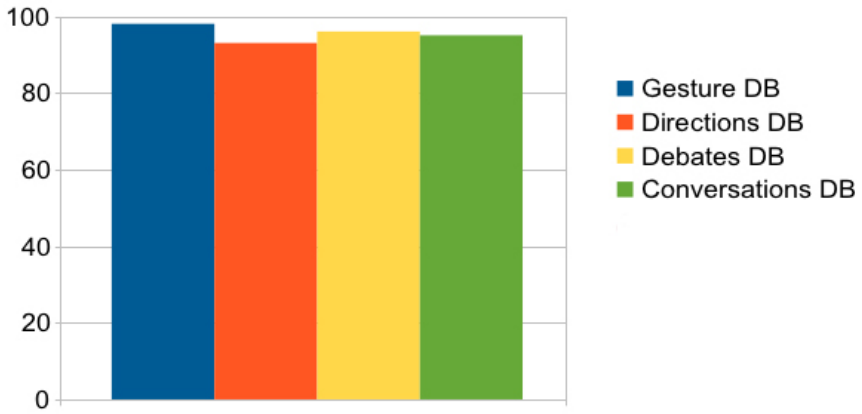

Figure 5.10: The results obtained from the motion phase estimation process when using different datasets.

Evaluating the Gesture Type Estimation

If one is given an input segment that belongs to a gesture phase, the system is responsible for estimating first the type of gesture rather than the most probable motion, as in Jörg et al. Jörg et al. (2012a), and then estimating the most appropriate motion. This is achieved by assigning the computed global features that describe each gesture of the character in a distance metric. Therefore, in this case, it is required the ability to evaluate the accuracy of the system and to estimate the correct gesture type. As in the previous evaluation process, it was considered the leave-one-out cross validation methodology. However, for cases that require a method to understand how each of the features reacts during the motion estimation process, an evaluation that computes the approximate correct motion is presented in Figure 5.11. Moreover, to understand how the addition of each feature in Equation 5.12 influences the motion estimation process, in Figure 5.12, presents the correctly estimated gestures when an additional feature is used.

To evaluate the complete methodology as presented in the previous sections, given a gesture segment, the system was trained without accounting for the features of the input segment, and the gesture type to which the input segment belonged was then computed. The class confusion matrix (Figure 5.13) represents the results that were obtained from the evaluation process. Based on those results, the following can be stated. First, the presented methodology is responsible for correctly estimating more of the relative finger gestures compared with the solution proposed by Jörg et al. Jörg et al. (2012a) that achieved to estimate on average 80% correctly each gesture. The presented methodology achieves an overall 90% correctness on the gestures, which is a significant improvement.

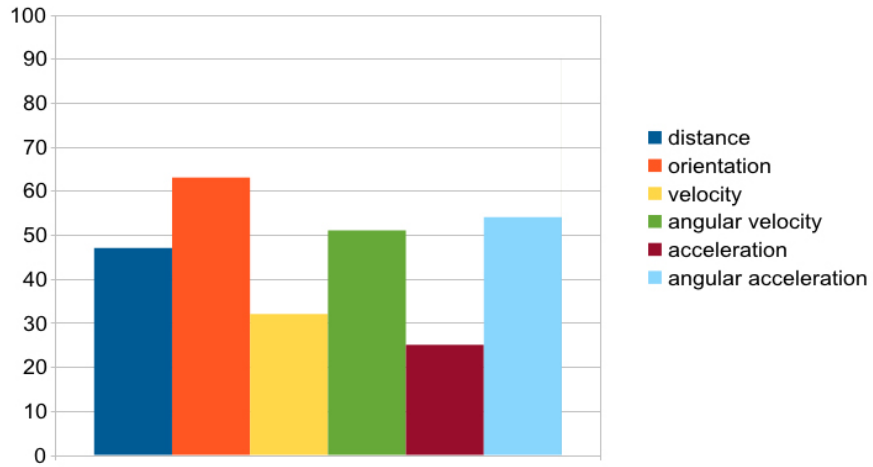

Figure 5.11: The correct estimation of the motion segments when using each of the examined features independently.

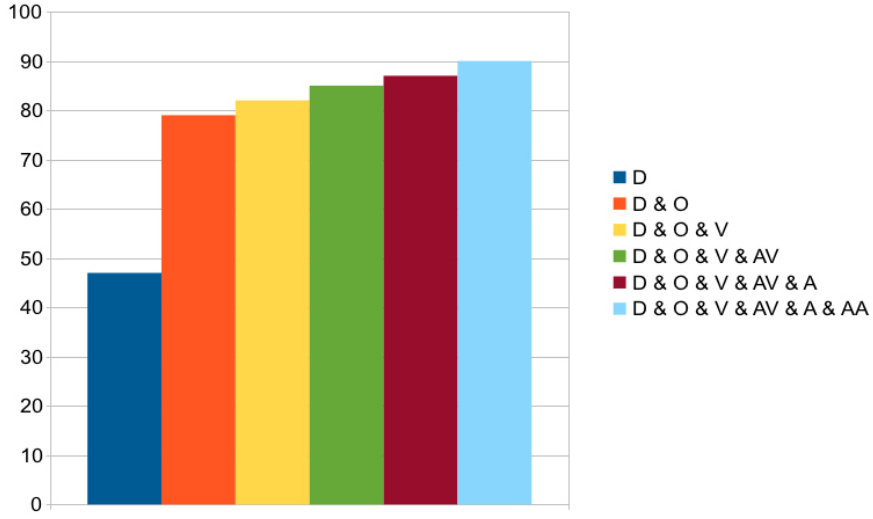

Figure 5.12: The influence of the motion estimation process when a new feature is added to Equation 5.12. D, O, V, AV, A, and AA denote the distance, orientation, velocity, angular velocity, acceleration, and angular acceleration motion features, respectively.

segment taken out of the database \ gesture type of the closest segments	att.	big	d-s	OK	PP	small	turn	wipe
att.	83	0	1	7	0	5	0	4
big	0	94	0	0	2	0	4	0
d-s	2	0	78	2	3	5	10	0
OK	6	0	0	91	0	0	0	3
PP	0	0	1	0	93	1	4	1
small	0	0	1	0	0	97	1	1
turn	0	1	1	0	2	0	93	3
wipe	0	1	0	0	2	2	5	90

Figure 5.13: The class confusion matrix shows the percentage of gestures that are estimated using the presented hybrid finger motion estimation process.

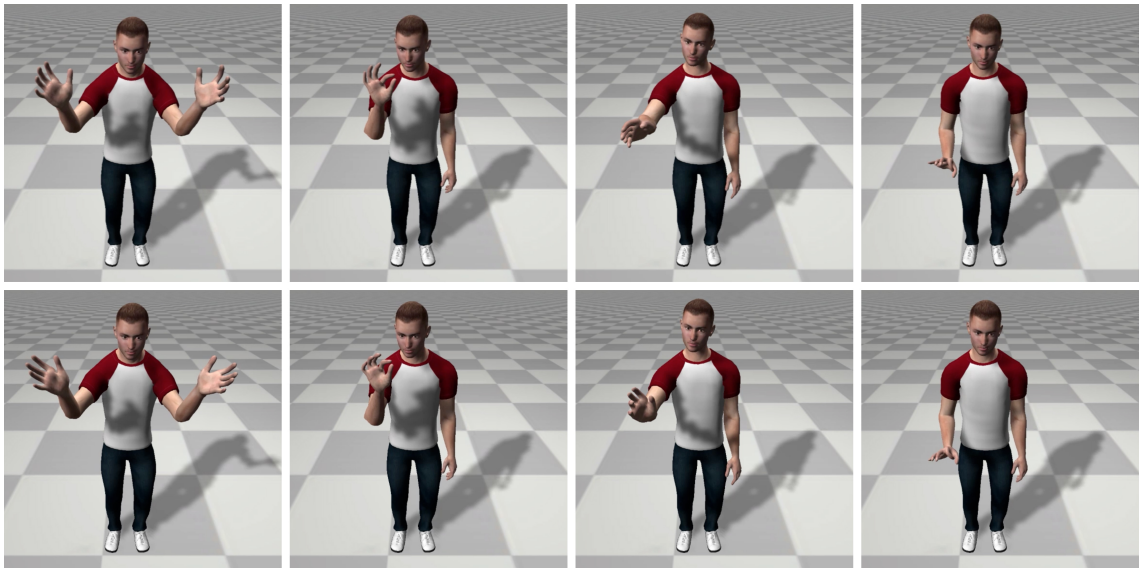

Figure 5.14: Given a reference motion (upper row), the system estimates and synthesises the most appropriate gesture of the character's finger (lower row).

Example synthesised postures of the character based on the presented methodology are illustrated in Figure 5.14.

5.3 Real-Time Finger Motion Synthesis

The previously presented off-line methodology is able to provide the required results for synthesising the motion of the characters fingers. Although, there are applications related to virtual reality that may require a real-time estimation of the character's fingers motion. Specifically, during the runtime of the application the system should be able to compute the necessary features of the input motion provided by the motion capture device. Based on the computed features, by using a motion feature distance function the most probable motions are retrieved. In this case, in order to avoid any discontinuities, as well as in order to retain the required meaning of the finger gesture, a transition cost function is implemented. Finally, having estimated the most probable motion, a transition process as it is proposed from the standard motion graphs methodology Kovar et al. (2002a) is implemented. The aforementioned steps that enclose the complete pipeline of the presented methodology are illustrated in Figure 5.15. The main difference of the pipeline presented below with the one presented in Figure 5.4 is its ability to estimate each motion segment by comparing one by one the motion features of the input and reference motion, instead of searching through a general representation of similar motion segments.

5.3.1 Motion Segmentation

By splitting the motion in long segments as in the previous off-line computation long delay between the input motion segment and the output synthesised motion will appear. Thus, to reduce this delay the presented methodology segments the motion into smaller sections. In this case, the followings conditions should be taken into account. Each pair of motion segments should be able to be blended efficiently without any discontinuity appearing. Additionally, the motion segments must be small enough in length, such as avoiding any long undesired delay during the runtime of the application. For these reasons, it was chosen to split the motion data in an equal length of segments that is 10 frames (0.33 seconds). The optimality of the length of the segment is ensured since a perceptual study Heron et al. (1974) indicated that such a delay could be affordable in visual presence. Therefore, while segmenting the motion data in small parts, both of the aforementioned conditions can be fulfilled. To conclude, for each dataset used, obtained from Jörg (2014), the resulted number of segments are shown in Table 5.1.

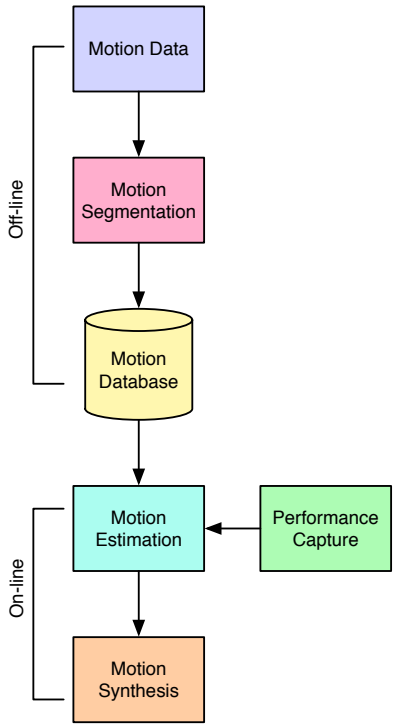

Figure 5.15: The pipeline of the presented real-time methodology. It consists on both off-line computations that segment and store the data in a database and the motion data and on-line computations for estimating and synthesising the motion of a character.

Dataset	Conversations	Debates	Directions	Gesture
Segments	1976	1864	1623	651

Table 5.1: The number of motion segments of each dataset that used.

5.3.2 Computing Motion Features

Based on previous methodologies that are related to gesture recognition, the gesture of the human can be estimated by the parameters provided by its wrist motion Yoon et al. (2001) Liu et al. (2004) Elmezain et al. (2008). Additionally, as shown by Jörg et al. Jörg et al. (2012a) the position and the orientation of the character's wrist enables estimation of high quality finger gestures. Although, in this case it is required a methodology that will be able to synthesise the desired motion of the character's fingers by using a commercial low-cost motion capture device such as the Microsoft's Kinect. The inability of computing the orientation of the performer's wrist gave the chance to search for various other motion features that could be used.

In general, the human's hand motion is able to provide various features. Among others, the most important is the temporal and the spatial extend, where both of these

can be computed for the character's wrist position and orientation, the velocity and the angular velocity, the acceleration and the angular acceleration, the curvature, and the height. Although, since any related feature to the character's wrist orientation cannot be computed from the motion capture device, it was considered to be computed the spatial extent $\bar{D}$, the velocity $\bar{V}$, the acceleration $\bar{A}$, the curvature $\bar{C}$, and the height $\bar{H}$ provided by the character's wrist. Therefore, for each of the segmented motion sequences it was considered to be computed the following equations:

$$\bar{D} = \log \frac{1}{n} \sum_{i=1}^{n} \|p_i - p_1\| \tag{5.14}$$

$$\bar{V}_i = \log \frac{1}{n} \sum_{i=1}^{n} \|v_i\| \tag{5.15}$$

$$\bar{A}_i = \log \frac{1}{n} \sum_{i=1}^{n} \|a_i\| \tag{5.16}$$

$$\bar{C}_i = \log \frac{1}{n} \sum_{i=1}^{n} \|v_i \times a_i\| \tag{5.17}$$

$$\bar{H}_i = \log \frac{1}{n} \sum_{i=1}^{n} \|h_i\| \tag{5.18}$$

where p_i, v_i, a_i, and h_i denote the position, velocity, acceleration, and the heigh of the character's wrist's at the $i-th$ frame of the motion segment. Having computed these features each of the motion segments s_i is represented as $s_i = \{\bar{D}_i, \bar{V}_i, \bar{A}_i, \bar{C}_i, \bar{H}_i\}$. It should be noted that each of the motion features is calculated according to the root position, velocity, acceleration and height.

5.3.3 Finger Motion Estimation

In the following two subsections it is firstly described the distance function that is used for estimating the most probable motion segments. Secondly, the ability to restrict the searching process by using a transition cost function is presented. The cost function is quite beneficial in the presented solution. It is achieves the required continuity between origin and a target motion segments, as well as the required meaning of the synthesised motion.

Distance Function

For retrieving a motion segment that is as close as possible to the features computed by the user's wrist, a distance function is required that should be able to estimate the most probable motions. Although, since a one to one mapping of the frames between the input

and the reference data, as it proposed by Jörg et al. Jörg et al. (2012a), occurs at a high computation cost, the motion estimation process is assigned to the features of the motion. Based on those features, it is now required to compute a distance function that will sort the motion segments according to cost. Therefore, the distance function that is implemented computes the following:

$$D_F = (D - \bar{D}_i)^2 + (V - \bar{V}_i)^2 + (A - \bar{A}_i)^2 + (C - \bar{C}_i)^2 + (H - \bar{H}_i)^2 \tag{5.19}$$

where D, V, A, C, and H is the spatial extent, the velocity, the acceleration, the curvature, and the height motion features of the input motion segment respectively. Based on this distance function, the system is able to rank the motion segments contained in the database. In this case, by assuming that the motion segment that has the lower cost is the most probable for the motion synthesis process, undesired effects such as jerkiness may appear during the transition process. This is since, the transition process may not be completed due to the length of the segments. Moreover, this assumption cannot be marked as correct since even if the motion features of the estimated motion fulfils as close as possible the motion features of the input motion segment, it is ensures neither the continuity between the origin and the target motion, nor the required meaning where the synthesised motion should have. For that reason, an additional computational step is required for retrieving the most probable motion, as it is presented below.

Transition Cost

Each motion segment contains a small representation of a gesture, making it difficult to calculate a long motion trajectory that retains the required meaning of the motion. In order to overcome this inability, a highly conditioned cost function is implemented. With the presented cost function the system estimates a motion that is as close as possible to the original motion. Therefore, achieving the required continuity between the last frame A to the first frame B of the origin and the target motion respectively. Thus, the following transition cost function is implemented:

$$\begin{aligned} c_T = &\frac{1}{m}\Big(\sum_{i=1}^{m}(P_A(i) - P_B(i))^2 + \sum_{i=1}^{m}(R_A(i) - R_B(i))^2 + \sum_{i=1}^{m}(V_A(i) - V_B(i))^2 \\ &+ \sum_{i=1}^{m}(AA_A(i) - AA_B(i))^2)\Big) \end{aligned} \tag{5.20}$$

where $P_A(i)$ and $P_B(i)$ are the positions of the $i-th$ degree of freedom of the finger joints in frame A and frame B, respectively, $R_A(i)$, and $R_B(i)$ are the corresponding orientation, $V_A(i)$ and $V_B(i)$ are the corresponding velocities, and, $AA_A(i)$ and $AA_B(i)$

are the corresponding angular velocities. Finally, m denotes the total degrees of freedom of the character's hand skeletal model that used, which is $m = 25$.

In this case, the most valid motion should be selected. Although, assigning a cost function at every motion segment, a high computational cost is incurred. For that reason, it was chosen to compute the cost function for the first twenty motion segments, for computational reasons, as those result from the motion features distance function. Therefore, the motion where the c_T is minimum is selected as the one that best matches the user's input.

5.3.4 Implementation

The presented methodology was implemented on an Intel i7 at 2.2 GHz with 8 GB memory. For the motion capture process the Microsoft's Kinect motion capture device Microsoft (2014) was used. During the runtime of the application the user perform actions related to the motions segments contained in the database. In this case, for achieving a real-time implementation a buffer-based methodology, which is responsible for storing the input poses of the performer was implemented. Specifically, since the length of each motion segment is 10 frames, which results after the segmentation process, an equal length buffer is implemented to store the input data. The input data is the positions and orientation of each joint of the character's upper-body for the last 10 frames. In order to avoid any discontinuity as well as in order to avoid any wrong synchronisation of the synthesised motion of the character, the motion of the user is displayed simultaneously with the computed motion of the character's fingers.

More specifically, while using this buffer-based approach the following should be mentioned. The motion estimation process starts while the buffer is filled with every ten new frames. Having computed the most probable motion of the character's fingers, this motion is assembled to the corresponding full-body motion of the character that is retrieved from the motion capture device. Thus, during the runtime of the application at each time step the first frame that stores the motion of the performers is displayed simultaneously with the first frame of the motion of the fingers. This procedure works iteratively until the user request to quit the application. A simple explanation of the methodology is shown in Figure 5.16. It should be noted that this buffer-based methodology produces an artificial delay since the resulted motion is displayed when the buffer is filled with ten new frames of data. Based on the results provided by a perceptual study conducted by Gulliver and Ghinea (2007), it was assumed that this delay does not affect the perception of the

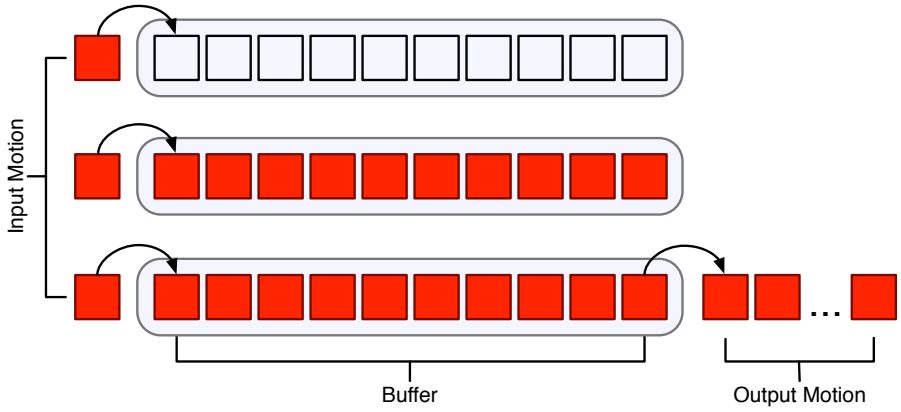

Figure 5.16: A simple representation of the buffering process of the captured motion data.

Figure 5.17: Example poses of the character's fingers with its associated body posture, synthesised with the presented solution while using the gesture (left), conversation (middle), and direction (right) dataset.

observers, since it is achieved the motion of the fingers to be displayed simultaneously and synchronously with the full-body motion of the character. Finally, since a smooth transition is required between every pair of corresponding motion sequences, the transition process, as it is described by the standard motion graphs methodology, which is proposed by Kovar et al. Kovar et al. (2002a) is implemented. Example postures synthesised with the presented methodology are illustrated in Figure 5.17.

In such a case, another possible solution could be developed. For example, for estimating the motion of the fingers in real-time a windowing function could provide the ability of removing the artificial delay produced by the proposed buffering process. However, disadvantages may appear. Firstly, while dealing with the analysed representation of the motion features the system is unable to ensure the continuity of a gesture. This can be omitted while dealing with the trajectories of the gesture. However, the disadvantage of such implementation is the high computational cost that will be required by the system to estimate and synthesise the motion of a character.

5.3.5 Evaluations and Results

In this section, the results obtained from the evaluation process of the presented methodology are presented. More specifically, the ability to synthesise the motion of the character's fingers by using the gesture dataset (8 gesture types $\times$ 8 repetitions), which can be found in Jörg (2014). For the evaluation process, firstly it was examined the ability of correctly estimating the motion of the character's fingers by using each of the motion features used separately. Moreover, in order to show the efficiency of the cost function that was used in the presented methodology, the results obtained while the cost function is constrained for the motion estimation process are presented.

In this case, the leave-one-out cross validation methodology was used. More specifically, for each of the gestures contained in the database, by leaving out one of those, and then it was computed the correct estimation of the gesture type. This evaluation iterated for each one of the 64 motion sequences contained in the dataset. Finally, it should be noted that during the evaluation process each gesture was performed in a controlled and systematic way, starting and ending in a rest pose with the arms hanging related at the side of the body. Each input reference motion was segmented in short segments equal to 10 frames as it was presented in Section 5.3.1. The evaluation steps are presented in the following subsections.

Motion Features Evaluation

Using motion features to represent short segments of motion is a key issue in the presented methodology. Therefore, an evaluation that indicates how each of the examined features enhances the motion estimation process was conducted. More specifically, in order to understand the efficiency of each of the motion features that were used, each one of those were evaluated separately. The results of this evaluation process are presented in Figure 5.18. By observing the results the following can be determined. Firstly, each of the features when it is used on its own provides a low estimation rate. It is assumed that this low rate depends on the inability of each feature to describe in an efficient way each one of the motion segments. Thus, while the motion segments consists on a small number of frames, intersection or quite close values may appear. Therefore, for a better estimation of each motion segment, it would be quite beneficial to examine the ability of combining different motion features.

In a second evaluation step, it was considered the ability of evaluating the accuracy of the presented methodology by adding a new feature every time. The results obtained

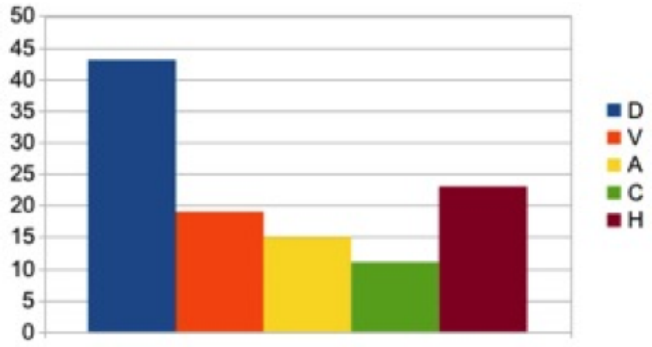

Figure 5.18: Percentage of correctly estimated finger gestures by using each of the features separately. D, V, A, C, H denote the spatial extent, the velocity, the acceleration, the curvature, and the height respectively.

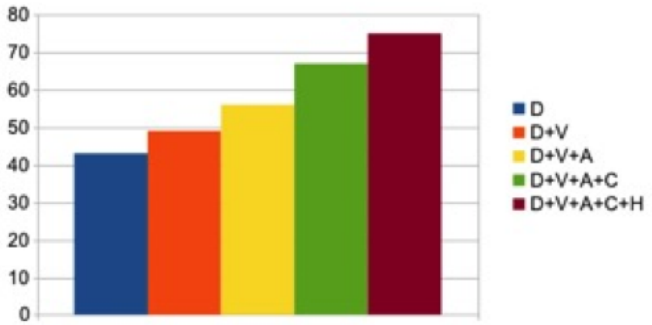

Figure 5.19: Percentage of correctly estimated finger gestures while any of the features is added to the motion estimation process. D, V, A, C, H denote the spatial extent, the velocity, the acceleration, the curvature, and the height respectively.

from this evaluation process are shown in Figure 5.19. Our initial assumption was that the more motion features are used the more accurate estimation can be performed by the system. This assumption is supported by this evaluation. While a new feature is added, the finger motion estimation rate grows, resulting in a highest value of 74%, while all the examined motion features are used. It should be noted that even if the order where the motion features are added changes, this increment is still appears. The presented methodology achieves results quite close to those provided from the methodology of Jörg et al. Jörg et al. (2012a), where as mentioned achieves to estimate on average 80% correct each gesture. Although, considering that the presented methodology is able to provide those results in real-time, it can be said that it is a significant improvement over the previous solutions.

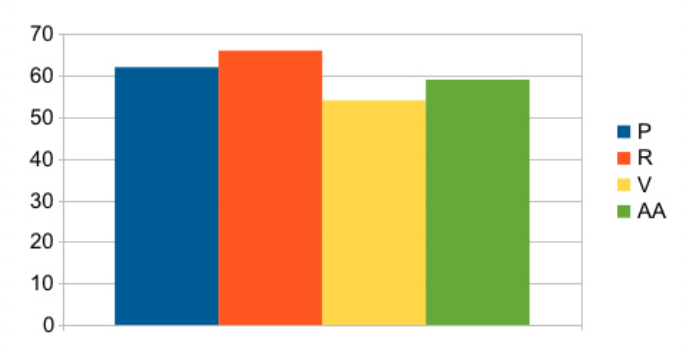

Figure 5.20: Percentage of correctly estimated finger gestures while each one of the features used in the cost function. P, R, V, and AA represent the position, orientation, velocity, and angular acceleration constraints respectively.

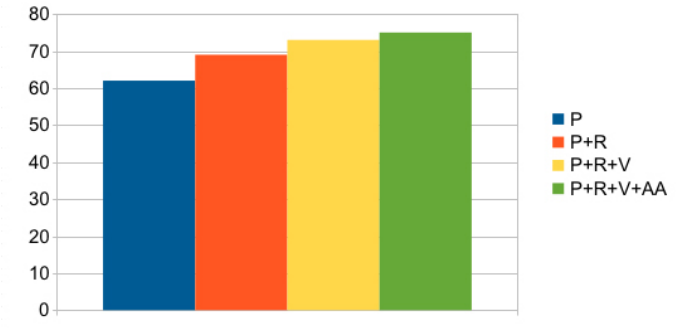

Figure 5.21: Percentage of correctly estimated finger gestures while a new feature is added in the cost function. P, R, V, and AA represent the position, orientation, velocity, and angular acceleration constraints respectively.

Cost Function Evaluation

Continuing the evaluation process of the presented methodology, it could be quite beneficial to indicate the need to use a cost function (see Equation 5.20) during the motion estimation process. In this case, firstly the estimation rate for each of the features was examined separately (see Figure 5.20), as well as when any of the motion features is added to the cost function (see Figure 5.21). By observing the results, it should be noted that any of the features in the presented cost function affect differently the estimation rate. Although, as it was assumed previously, the more motion features are used, the more accurate an estimation can be. Therefore, as the results shown by assigning a highly conditional cost function allows the system to estimate in a better way the most probable motion of the character. Finally, it should be noted that an additional advantage of the presented cost function is the ability to ensure the continuity of a gesture. Thus, the system is able to synthesise meaningful phases of finger gestures.

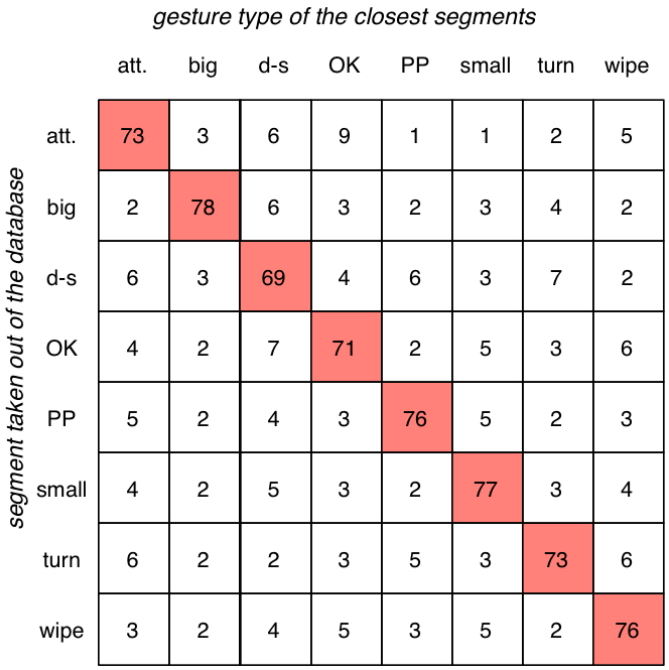

Figure 5.22: The confusion matrix that shows the percentage of correct estimation of each gesture type, as it is resulted by the presented methodology.

Evaluating Gesture Estimation

By examining each of the gestures that were used in the evaluation process, a confusion matrix was generated. Figure 5.22 shown the correct estimation of each gesture type that results from the presented solution. In observing these results it should be noted that in comparison with the methodology presented by Jörg et al. Jörg et al. (2012a), the generalisation process that is used for representing the motion segments provides a quite close distribution of the results. Therefore, it is difficult to say that such a generalisation process is able to describe efficiently a specific gesture type. On the other hand, while the results that are obtained for each gesture type are quite close, the following can be assumed. While using more motion features, to describe each of the motion segments, the results may increase equally, providing a higher estimation rate.

5.4 Discussion

In this chapter, two different methodologies for estimating the motion of the character's finger were presented. In the first methodology, the motion estimation process was achieved by analysing different finger gestures, and by assigning the motion estimation process to a hybrid estimation method. This hybrid method, instead of searching one by one the motion segments searches firstly for the generalised representation of each gesture type,

and then searches for the most appropriate motion. Therefore, the advantage of such a solution is firstly the decrease of the computation cost that is required, and secondly the increment of the correct estimation rate as demonstrated in Section 5.2.4.

Additionally, by using motion features to describe each motion segment it was possible to compute in real-time the motion of the character's fingers, during the performance capture process. However, a basic limitation that appeared during the developing process is the required continuity for achieving the desired meaning of a finger gesture. Thus, by designing a highly constrained cost function it was possible to synthesise finger motion sequences that keeps the required meaning. It should be noted that the estimation rate, compared with the off-line methodology is lower, although, since the method achieves this in real-time it can be said that it is a clear improvement over existing solutions.

Chapter 6

Conclusions and Future Work

Animating virtual characters is a challenging research area. This is since, depending on the methodology that is used for synthesising the motion of a character i.e., kinematics, data-driven, physics (see Section 2), the required level of realism may not appear in the final synthesised motion. For the data-driven techniques, methodologies for handling, estimating or retrieving, and finally synthesising the motion data that keeps the required realism are required.

In this thesis three different methodologies for character animation were presented. These methodologies are incorporated in a character animation framework that is responsible for enhancing the naturalness of the final synthesised motion. Specifically, the enhancement of the realism of a new synthesised motion is achieved incrementally by the use of each of the presented motion synthesis techniques.

Firstly, by incorporating measurements from real human locomotion, a novel methodology that synthesises in a human-like way the motion of the virtual character was presented. Specifically, in this methodology the transition process where real humans perform while they transform from a locomotion behaviour to any other was measured. Then, by assigning this action transition process to transition graphs, as well as by assigning the different locomotion actions of the character to footprint patterns the system achieves the synthesis of the required motion.

However, because the generated animation is based only on the ability to extract the most suitable motions from a collection in a database, the results always depend on these motions. Therefore, in the future, implementation techniques related to kinematics and biomechanics could be used to enrich the number of generated motions. In addition, the generated motion is based on motion capture data and it is assumed to be physically correct. Another key issue that should be investigated in a future implementation is

the ability to examine dynamics or generate a motion model in which physics-based constraints, such as balancing and energy minimization are considered. Finally, the current implementation is responsible for synthesising the motion of the character by interpreting the information retrieved from the footprint patters to produce a limited number of synthesised motions and ideally this number of motions could be increased. A simple example over those already covered is the ability of generating running action with short strides and walking actions with long strides. In general, this can easily be implemented by assigning the desired action of the character at every footprint. Although, the application will lose its initial advantage, which is the automatical generating of the desired locomotion by recognising each particular action based on the footprints. It should be stated that the possibility of automatically generating actions for multiple motions could be a further area of research.

In the second motion synthesis techniques, it was examined the ability of synthesising style variations of an input motion during the performance capture process. The synthesis of the required style content during the runtime of the application can be quite beneficial in film production and other visual effects scenarios. This is since, instead of capturing repeating motion sequences that contain the required style content, it is possible to retrieve the style content from a motion sequence and then to apply this style content to a different motion. In the presented methodology this is achieved by developing a simple correlation between two different behaviours, which then is used as a parameter to a MAP framework. Generally, the advantage of such a methodology is the elimination of the post processing that is required for synthesising the required style. The results obtained from such a methodology shown that the presented approach provides natural-looking results. Although it was not possible to adjust the stylistic content during the runtime. Another limitation of the presented method is the inability to handle timing variations for each stylistic motion. Therefore, the user must mimic the timing variations to synthesise a natural looking representation of the style component. This problem can potentially be overcome by synthesising a motion model that, in conjunction with the stylistic motion synthesis process, will maintain the timing variations for each individual motion. The user will thus not be responsible for timing a specified style. Rather, the system will automatically adjust the timing variations. Therefore, future plans employ the ability to design a high-quality stylistic motion mixer between different stylistic behaviours that can produce a virtual character. Thus, a user can adjust the stylistic content of the performer in real-time to perform synthesis on-the-fly for different stylistic behaviours. It is postulated

that such a motion synthesis process would have significant application in the film and animation industries because the style content of the motion could be adjusted during the capturing process, thereby providing the ability to produce the desired stylistic behaviour for the character without repeatedly capturing the same motion sequence.

The final motion synthesis techniques that were presented are based upon the ability of synthesising the motion of the character's fingers. As have been stated in perceptual studies, such as Hoyet et al. (2012) Jörg et al. (2010) Jörg et al. (2012b), the realism and the meaning of a motion sequence can be increased when details, such as the motion of the character's fingers, are added to the final synthesised motion. The advantages of such a methodology is based upon the ability to assign the motion searching process to motion features instead of comparing long trajectories. Therefore, a reduction of the computation time was achieved.

Such a motion searching process can be quite beneficial in cases that require a real-time finger motion estimation and synthesis process. As presented, the real-time implementation is achieved by firstly segmenting the motion data in short length segments and by implementing a buffering step of the input motion data. Specifically, the input motion is stored in a buffer, and the system searches to find the most close motion segment while the buffer is filled with the new frames. During the synthesis process, a basic limitation of the system was its ability to synthesise the required continuity, as well as to synthesise in an effectively the meaningful phases of the finger gestures. This problem was overcome by developing a highly constrained cost function that was responsible for retrieving the motion segment that minimises the transition cost between the displayed (origin) and the next (target) segment. Thus, the system was able to synthesise the required motion, although, having a delay equal to the buffer's size. Considering that there are various applications related to virtual reality, such as the telepresence, that require highly detailed animated characters, it would be quite beneficial to extend the proposed solution by developing a motion estimation methodology by using techniques related to hidden Markov models, which have been extensively used in speech and gesture recognition.

The motion estimation process is always dependent on the features that are used. Therefore, in future work it could be quite interesting to collect and evaluate a variety of features that can be computed from the existing motion data. It is assumed that for achieving highly accurate results an understanding of how each feature reacts during the estimation process is required. Additionally, since a single feature is not able to estimate directly the most valid motion, it is assumed that an evaluation of different combinations of

the motion features may provide quite useful results. Therefore, trying to find an optimal feature set, the aforementioned evaluation is likely to be effective.

To conclude, motion synthesis techniques vary as the requirements for animating a virtual character vary. Different methodologies for animating virtual characters are always required for improving not only the naturalness of the synthesised motion but also for eliminating the post processing and for decreasing the computational time. Thus, this research makes several key contributions to the realism of synthesised characters and suggests various other character animation methodologies that have scope for re-examination. These methods should be able to provide novel ways where the motion data is handled, parameterised and synthesised. Moreover, these methods should be able to synthesise the desired motions in real-time. The real-time implementation being key in the entertainment industry where it will provide the ability to increase the realism of the animated character, and result in enhanced immersion of the user. Therefore, it is not only desired to produce animated characters that act as humans, but also animated characters that act as humans in real-time. The contributions of the presented research are summarised below:

- A methodology for synthesising the locomotion of a virtual character based on predefined footprints. For the motion synthesis process the footprint patterns are responsible for providing the necessary parameters for synthesising the required locomotion behaviour of the character. Moreover, a behaviour-to-behaviour transition process ensures more natural representation of the character's motion.

- A methodology for animating virtual characters with style variations during the motion capture process. The method takes the benefit of building a correlation between existing motion styles. Thus, the system is responsible for animating the virtual character while it keeps the desired stylistic content.

- A novel hybrid searching methodology for the finger motion estimation process. This method is based on searching the most similar motions based on motion features. Additionally, with some key modifications a real-time implementation is achieved.

References

Agarwal, A. and Triggs, B. (2005). Monocular human motion capture with a mixture of regressors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, page 72. IEEE Press. 33

Alberts, B. (2007). *Molecular Biology of the Cell*. Garland Science Publishing. 2

Alexa, M. and Müller, W. (2000). Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418. 32

Allen, B., Curless, B., and Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics*, 22(3):587–594. 12

Amaya, K., Bruderlin, A., , and Calvert, T. (1996). Emotion from motion. In *Graphics Interface*, pages 222–229. ACM Press. 26

Aristidou, A. and Lasenby, J. (2009). nverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. Technical report, University of Cambridge. 23

Assus (accessed 30/5/2014). Xtion motion capture device. `http://www.asus.com/Multimedia/Xtion_PRO/`. 4

Autodesk (accessed 30/5/2014a). Autodesk maya. `http://www.autodesk.com/products/autodesk-maya/`. ix, 8, 21

Autodesk (accessed 30/5/2014b). Autodesk motionbuilder. `http://www.autodesk.com/products/motionbuilder/`. 8, 21

Badler, N., Hollick, M., and Granieri, J. (1993). Realtime control of a virtual human using minimal sensors. *Presence: Teleoperators and Virtual Environments*, 2(1):82–86. 37

Baker, D. and Wampler, C. (1988). On the inverse kinematics of redundant manipulators. *The International journal of robotics research*, 7(2):3–21. 22

Baran, I. and Popović, J. (2007). Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics*, 26(3):Article No. 72. 12

Bindiganavale, R. and Badler, N. I. (1998). Motion abstraction and mapping with spatial constraints. In *Modelling and Motion Capture Techniques for Virtual Environments*, pages 70–82. Springer Berlin Heidelberg. 12

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer. 73

Blender (accessed 30/5/2014). Blender 3d. `http://www.blender.org/`. 8, 21

Boulic, R. and Raunhardt, D. (2009). Integrated analytic and linearized inverse kinematics for precise full body interactions. In *Motion in Games*, pages 231–242. Springer Berlin Heidelberg. 37

Boulic, R., Thalmann, D., and Magnenat-Thalmann, N. (1990). A global human walking model with real time kinematic personification. *The Visual Computer*, 6(6):344–358. 23

Brand, M. and Hertzmann, A. (2000). Style machines. In *27th annual conference on Computer graphics and interactive techniques*, pages 183–192, New York, USA. ACM Press. 15, 32, 33, 40

Bruderlin, A. and Calvert, T. (1993). Interactive animation of personalized human locomotion. In *Graphics Interface*, pages 17–23. ACM Press. 24

Bruderlin, A. and Calvert, T. (1996). Knowledge-driven, interactive animation of human running. In *Graphics Interface*, pages 213–221. 24

Bruderlin, A. and Calvert, T. W. (1989). Goal-directed, dynamic animation of human walking. *ACM SIGGRAPH Computer Graphics*, 23(3):233–242. 23

Bruderlin, A. and Williams, L. (1995). Motion signal processing. In *22nd annual conference on Computer graphics and interactive techniques*, pages 97–104. ACM Press. 26, 27, 41

Buhmann, M. D. (2003). *Radial basis functions: theory and implementations*, volume 12. Cambridge university press. 69

Cao, Y., Faloutsos, P., and Pighin, F. (2003). Unsupervised learning for speech motion editing. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 225–231. Eurographics Association. 33, 41

Carnegie Mellon University (accessed 30/5/2014). Motion capture database. `http://mocap.cs.cmu.edu/`. 68, 75

Carvalho, S. R., Boulic, R., and Thalmann, D. (2007). Interactive lowdimensional human motion synthesis by combining motion models and pik. *Computer Animation and Virtual Worlds*, 18(4–5):493–503. 11

Carvalho, S. R., Boulic, R., and Thalmann, D. (2009). Motion pattern encapsulation for data-driven constraint-based motion editing. In *Motion in Games*, pages 116–127. Springer Berlin Heidelberg. 33

Carvalho, S. R., Boulic, R., Vidal, C. A., and Thalmann, D. (2013). Latent motion spaces for full-body motion editing. *The Visual Computer*, 29(3):171–188. x, 11, 27, 33

Chai, J. and Hodgins, J. K. (2005). Performance animation from low-dimensional control signals. *ACM Transactions on Graphics*, 24(3):686–696. 15, 38, 39, 40

Chenney, S., Pingel, M., Iverson, R., and Szymanski, M. (2002). Simulating cartoon style animation. In *2nd international symposium on Non-photorealistic animation and rendering*, pages 133–138, New York, USA. ACM Press. 1

Choi, M. G., Lee, J., and Shin, S. Y. (2003). Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2):182–203. 34, 35, 46

Chung, S. and Hahn, J. (1999a). Animation of human walking in virtual environments. In *Computer Animation*, pages 4–15. ix, 23, 24

Chung, S. and Hahn, J. (1999b). Animation of human walking in virtual environments. In *Computer Animation*, pages 4–15. 35

Coros, S., Beaudoin, P., Yin, K. K., and van de Pann, M. (2008). Synthesis of constrained walking skills. *ACM Transactions on Graphics*, 27(5):Article No. 113. 34

Deng, Z., Chiang, P. Y., Fox, P., and Neumann, U. (2006). Animating blendshape faces by cross-mapping motion capture data. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 43–48. 70

Derouet-Jourdan, A., Bertails-Descoubes, F., Daviet, G., and Thollot, J. (2013). Inverse dynamic hair modeling with frictional contact. *ACM Transactions on Graphics*, 32(6):Article No. 159. 25

Deutscher, J., Blake, A., and Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 126–133, New York, USA. IEEE Press. 12

Deutscher, J. and Reid, I. (2005). Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205. 12

Egges, A. and Van Basten, B. (2010). One step at a time: animating virtual characters based on foot placement. *The Visual Computer*, 26(6):497–503. 34, 46, 60

Elgammal, A. and Lee, C. (2004). Separating style and content on a nonlinear manifold. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–485. 40

ElKoura, G. and Singh, K. (2003). Handrix: animating the human hand. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 110–119. Eurographics Association. 43

Elmezain, M., Al-Hamadi, A., and Michaelis, B. (2008). Real-time capable system for hand gesture recognition using hidden markov models in stereo color image sequences. *Journal of WSCG*, 16(1-3):65–72. 81, 100

Faloutsos, P., Van de Panne, M., and Terzopoulos, D. (2001a). Composable controllers for physics-based character animation. In *28th annual conference on Computer graphics and interactive techniques*, pages 251–260. ACM Press. 25

Faloutsos, P., van de Panne, M., and Terzopoulos, D. (2001b). The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computer & Graphics*, 25(6):933–953. ix, 25

Fod, A., Matarić, M. J., and Jenkins, O. C. (2002). Automated derivation of primitives for movement classification. *Autonomous robots*, 12(1):39–54. 86

Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631. 73

Girard, M. and Maciejewski, A. A. (1985). Computational modeling for the computer animation of legged figures. *ACM SIGGRAPH Computer Graphics*, 19(3):263–270. 25

Glardon, P. (2006). *On-line locomotion synthesis for virtual humans*. PhD thesis, EPFL. 20, 22, 30

Glardon, P., Boulic, R., and Thalmann, D. (2005). On-line adapted transition between locomotion and jump. In *Computer Graphics International*, pages 44–50. IEEE Press. 35

Gleicher, M. (2001). Comparing constraint-based motion editing methods. *Graphical models*, 63(2):107–134. 11

Greg, R. (2007). *Inside the human body: using scientific and exponential notation*. The Rosen Publishing Group. 2

Grochow, K., Martin, S. L., Hertzmann, A., and Popović, Z. (2004). Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531. 32, 33

Gulliver, S. R. and Ghinea, G. (2007). The perceptual and attentive impact of delay and jitter in multimedia delivery. *IEEE Transactions on Broadcasting*, 53(2):449–458. 103

Ha, D. and Han, J. (2008). Motion synthesis with decoupled parameterization. *The Visual Computer*, 24(7-9):587–594. x, 28, 30, 43

Ha, S., Bai, Y., and Liu, C. K. (2011). Human motion reconstruction from force sensors. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 129–138. Eurographics Association. 37, 38

Hardy, R. L. (1990). Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968–1988. *Computers & Mathematics with Applications*, 19(8):163–208. 69

Heron, J. R., Regan, D., and Milner, B. A. (1974). Delay in visual perception in unilateral optic atrophy after retrobulbar neuritis. *Brain*, 97(1):69–78. 99

Hodgins, J. K. and Wooten, W. L. (1998). Animating human athletes. In *Robotics Research*, pages 356–367. Springer London. 25

Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642. 29

Hoyet, L., Ryall, K., McDonnell, R., and O'Sullivan, C. (2012). Sleight of hand: perception of finger motion from reduced marker sets. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 79–86. ACM Press. 15, 112

Hsu, E., Pulli, K., and Popović, J. (2005). Style translation for human motion. *ACM Transactions on Graphics*, 24(3):1082–1089. 15, 29, 41

Huang, Y. and Kallmann, M. (2010a). Motion parameterization with inverse blending. In *Motion in Games*, pages 242–253. Springer Berlin Heidelberg. 11

Huang, Y. and Kallmann, M. (2010b). Motion parameterization with inverse blending. In *International Conference on Motion in Games*, pages 242–253. Springer Berlin Heidelberg. 34, 46

Ikemoto, L. and Forsyth, D. A. (2004). Enriching a motion collection by transplanting limbs. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108. Eurographics Association. 28, 43

Ishigaki, S., White, T., Zordan, V. B., and Liu, C. K. (2009). Performance-based control interface for character animation. *ACM Transactions on Graphics*, 28(3):Article No. 61. 37, 38

Jin, G. and Hahn, J. (2005). Adding hand motion to the motion capture based character animation. In *Advances in Visual Computing*, pages 17–24. Springer Berlin Heidelberg. 42

Jörg, S. (accessed 30/5/2014). Finger motion database. `http://people.cs.clemson.edu/~sjoerg/fms.html`. 86, 95, 99, 105

Jörg, S., Hodgins, J., and O'Sullivan, C. (2010). The perception of finger motions. In *Symposium on Applied Perception in Graphics and Visualization*, pages 129–133. ACM Press. 15, 79, 112

Jörg, S., Hodgins, J., and Safonova, A. (2012a). Data-driven finger motion synthesis for gesturing characters. *ACM Transactions on Graphics*, 31(6):Article No. 189. xii, 42, 43, 81, 82, 83, 84, 86, 87, 88, 94, 96, 100, 102, 106, 108

Jörg, S., Hodgins, J., and Safonova, A. (2012b). Data-driven finger motion synthesis for gesturing characters. *ACM Transactions on Graphics*, 31(6):Article No. 189. 15, 112

Jung, M. and Na, K. (2004). Hierarchical retargetting of fine facial motions. *Computer Graphics Forum*, 23(9):687–695. 69

Kallmann, M., Aubel, A., Abaci, T., and Thalmann, D. (2003). Planning collisionfree reaching motions for interactive object manipulation and grasping. *Computer Graphics Forum*, 22(3):313–322. 11

Kalra, P., Magnenat-Thalmann, N., Moccozet, L., Sannier, G., Aubel, A., and Thalmann, D. (1998). Real-time animation of realistic virtual humans. *IEEE Computer Graphics and Applications*, 18(5):42–56. 80

Kang, C., Wheatland, N., Neff, M., and Zordan, V. (2012). Automatic hand-over animation for free-hand motions from low resolution input. In *Motion in Games*, pages 244–253. Springer Berlin Heidelberg. 42, 44

Kendon, A. (2004). *Gesture: Visible action as utterance*. Cambridge University Press. 79

Kim, D., Koh, W., Narain, R., Fatahalian, K., Treuille, A., and O'Brien, J. F. (2013). Near-exhaustive precomputation of secondary cloth effects. *ACM Transactions on Graphics*, 32(4):Article No. 87. 25

Kim, J., Seol, Y., and Lee, J. (2012). Realtime performance animation using sparse 3d motion sensors. In *Motion in Games*, pages 31–42. Springer Berlin Heidelberg. 38, 39

Komura, T., Kuroda, A., Kudoh, S., Lan, T., and Shinagawa, Y. (2003). An inverse kinematics method for 3d figures with motion data. In *Computer Graphics International*, pages 266–271. 23

Kovar, L. and Gleicher, M. (2003). Flexible automatic motion blending with registration curves. In *ACM SIGGRAPH/Eurographics symposium on Computer animatio*, pages 214–224. Eurographics Association. 27, 29, 66

Kovar, L. and Gleicher, M. (2004). Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568. 11

Kovar, L., Gleicher, M., and Pighin, F. (2002a). Motion graphs. *ACM transactions on graphics*, 21(3):473–482. 28, 29, 42, 59, 83, 99, 104

Kovar, L., Schreiner, J., and Gleicher, M. (2002b). Footskate cleanup for motion capture editing. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104. Eurographics Association. 11, 24, 50

Kry, P. G. and Pai, D. K. (2006). Interaction capture and synthesis. *ACM Transactions on Graphics*, 25(3):872–880. 43

Laszlo, J., van de Panne, M., and Fiume, E. (1996). Limit cycle control and its application to the animation of balancing and walking. In *23rd annual conference on Computer graphics and interactive technique*, pages 155–162. 25

Lau, M., Chai, J., Xu, Y. Q., and Shum, H. Y. (2007). Face poser: interactive modeling of 3d facial expressions using model prior. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 161–170. Eurographics Association. 33

Levine, S., Theobalt, C., and Koltun, V. (2009). Real-time prosody-driven synthesis of body language. *ACM Transactions on Graphics*, 28(5):Article No. 172. 84

Li, H. (2010). *Animation Reconstruction of Deformable Surfaces*. PhD thesis, ETH Zurich. 13

Li, H., Adams, B., Guibas, L. J., and Pauly, M. (2009). Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics*, 28(5):Article No. 175. 12

Lim, I. S. and Thalmann, D. (2002). Construction of animation models out of captured data. In *IEEE International Conference on Multimedia and Expo*, pages 829–832. IEEE Press. 28

Liu, C. K., Hertzmann, A., and Popović, Z. (2005). Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics*, 24(3):1071–1081. 25, 40

Liu, C. K. and Zordan, V. B. (2011). Natural user interface for physics-based character animation. In *Motion in Games*, pages 1–14. Springer Berlin Heidelberg. 37

Liu, H., He, F., Cai, X., Chen, X., and Chen, Z. (2011a). Performance-based control interfaces using mixture of factor analyzers. *The Visual Computer*, 27(6-8):595–603. 39, 40

Liu, H., Wei, X., Chai, J., Ha, I., and Rhee, T. (2011b). Realtime human motion control with a small number of inertial sensors. In *Symposium on Interactive 3D Graphics and Games*, pages 133–140. ACM Press. 15, 38, 39, 40

Liu, N., Lovell, B. C., Kootsookos, P. J., and Davis, R. I. (2004). Model structure selection & training algorithms for an hmm gesture recognition system. In *International Workshop on Frontiers in Handwriting Recognition*. IEEE Press. 81, 100

Lou, H. and Chai, J. (2010). Example-based human motion denoising. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):870–879. 11

Lyard, E. and Magnenat-Thalmann, N. (2007). A simple footskate removal method for virtual reality applications. *The Visual Computer*, 23(9–11):689–695. 11

Magnenat-Thalmann, N. and Thalmann, D. (2005). *Handbook of virtual humans*. John Wiley & Sons. 4

Majkowska, A., Zordan, V. B., and Faloutsos, P. (2006). Automatic splicing for hand and body animations. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 309–316. 42

McLachlan, G. and Krishnan, T. (2007). *The EM algorithm and extensions*, volume 382. John Wiley and Sons. 73

Meredith, M. and Maddock, S. (2001). Motion capture file formats explained. Department of Computer Science, University of Sheffielf. 7

Metamotion (accessed 30/5/2014). Motion capture suits. `http://www.metamotion.com/`. ix, 8

Microsoft (accessed 30/5/2014). Kinect motion capture system. `http://www.microsoft.com/en-us/kinectforwindows/`. 4, 73, 103

Miller, N., Jenkins, O. C., Kallmann, M., and Mataric, M. J. (2004). Motion capture from inertial sensing for untethered humanoid teleoperation. In *4th IEEE/RAS International Conference on Humanoid Robots*, pages 547–565, New York, USA. IEEE Press. 5

Monzani, J. S., Baerlocher, P., Boulic, R., and Thalmann, D. (2000). Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum*, 19(3):11–19. 12

MPI Informatik (accessed 30/5/2014). Motion capture database hdm05. `http://resources.mpi-inf.mpg.de/HDM05/`. 6

Mukai, T. and Kuriyama, S. (2005). Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070. 11, 28, 32

Multon, F., France, L., Cani, M.-P., and Debunne, G. (1999). Computer animation of human walking: a survey. *Journal of Visualization and Computer Animation*, 10:39–54. 22

Neff, M. and Fiume, E. (2006). Methods for exploring expressive stance. *Graphical Models*, 68(2):133–157. 25, 41

Neff, M. and Seidel, H. P. (2006). Modeling relaxed hand shape for character animation. In *Articulated Motion and Deformable Objects*, pages 262–270. Springer Berlin Heidelberg. 43

Ng, W. W., Choy, C. S., Lun, D. P., and Chau, L. P. (2010). ynchronized partial-body motion graphs. In *ACM SIGGRAPH ASIA Sketches*, page 28. ACM Press. 29

Nguyen, N., Wheatland, N., Brown, D., Parise, B., Liu, C. K., and Zordan, V. (2010). Performance capture with physical interaction. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 189–195. Eurographics Association. 37, 38

Noh, J. and Neumann, U. (2001). Expression cloning. In *28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 277–288. ACM Press. 69

Organic Motion (accessed 30/5/2014). Motion capture system. `http://www.organicmotion.com/`. 4

Osipa, J. (2010). *Stop staring: facial modeling and animation done right*. John Wiley and Sons. 64

Pan, J., Yang, X., Xie, X., Willis, P., and Zhang, J. J. (2009). Automatic rigging for animation characters with 3d silhouette. *Computer Animation and Virtual Worlds*, 20(2–3):121–131. 12

Park, S. I., Shin, H. J., Kim, T. H., and Shin, S. Y. (2004.). On-line motion blending for real-time locomotion generation. *Computer Animation and Virtual Worlds*, 15(3–4):125–138. 28

Pausch, R., Snoddy, J., Taylor, R., Watson, S., and Haseltine, E. (1996). Disney's aladdin: First steps toward storytelling in virtual reality. In *23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 193–203, New York, USA. ACM Press. 1

Perlin, K. (1995). Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15. 27, 41

Plasticboy, T. R. (accessed 30/5/2014). Skeletal system 3d model. `http://therealplasticboy.deviantart.com/art/Skeletal-System-3D-Model-03-147332063`. ix, 3

Raunhardt, D. and Boulic, R. (2009). Motion constraint. *The Visual Computer*, 25(5–7):509–518. 11

Reallusion (accessed 30/5/2014). iclone 3d animation tool. `http://www.reallusion.com/iclone/`. 21

Rose, C., Cohen, M. F., and Bodenheimer, B. (1998). Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40. ix, 28, 29

Samadani, A. A., DeHart, B. J., Robinson, K., Kulic, D., Kubica, E., and Gorbet, R. (2011a). A study of human performance in recognizing expressive hand movements. In *IEEE RO-MAN*, pages 93–100. 15

Samadani, A. A., DeHart, B. J., Robinson, K., Kulic, D., Kubica, E., and Gorbet, R. (2011b). A study of human performance in recognizing expressive hand movements. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 93–100. 79

Savannah College (accessed 30/5/2014). Savannah college of art and design. `http://www.sfdm.scad.edu/intranet/students/cage/eqpages/mocapsuit.html`. ix, 7

Schwartz, M. H. and Rozumalski, A. (2005). A new method for estimating joint parameters from motion data. *Journal of biomechanics*, 38(1):107–116. 29

Semwal, S., Hightower, R., and Stansfield, S. (1998). Mapping algorithms for real-time control of an avatar using eight sensors. *Presence: Teleoperators and Virtual Environments*, 7(1):1–21. 37

Shapiro, A., Cao, Y., and Faloutsos, P. (2006). Style components. In *Graphics Interface*, pages 33–39. ACM Press. 41

Shapiro, A., Pighin, F., and Faloutsos, P. (2003). Hybrid control for interactive character animation. In *Pacific Conference on Computer Graphics and Applications*, pages 455–461. IEEE Press. 25

Shin, H. J., Lee, J., Shin, S. Y., and Gleicher, M. (2001). Computer puppetry: An importance-based approach. *ACM Transactions on Graphics*, 20(2):67–94. 36, 80

Shin, H. J. Lee, J. (2006). Motion synthesis and editing in lowdimensional spaces. *Computer Animation and Virtual Worlds*, 17(34):219–227. 11

Shiratori, T., Coley, B., Cham, R., and Hodgins, J. K. (2009). Simulating balance recovery responses to trips based on biomechanical principles. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 37–46. Eurographics Association. 1

Shiratori, T. and Hodgins, J. K. (2008). Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics*, 27(5):Article No. 123. 38

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1297–1304. IEEE Computer Society. 73

Shum, H. and Ho, E. S. (2012). Real-time physical modelling of character movements with microsoft kinect. In *ACM symposium on Virtual reality software and technology*, pages 17–24. ACM Press. 37, 38

SmithMicro (accessed 30/5/2014). Poser. `http://poser.smithmicro.com/`. 21

Steuer, J. (1992). Defining virtual reality: Dimensions determining telepresence. *Journal of communication*, 42(4):73–93. 1

Sturman, D. (1985). Interactive keyframe animation of 3d articulated models. Course Note, SIGGRAPH Course Number 10, Computer Animation: 3D Motion Specification and Control. 20

Sturman, D. J. (1998). Computer puppetry. *IEEE Computer Graphics and Applications*, 18(1):38–45. 36, 80

Sun, H. and Metaxas, D. (2001). Automating gait generation. In *28th annual conference on Computer graphics and interactive techniques*, pages 261–270. ACM Press. 23

Szécsi, L. and Szirmay-Kalos, L. (2010). General purpose computing on graphics processing units. In Iványi, A., editor, *Algorithms of Informatics*, pages 1451–1495. Mond-Art Kiadó. 69

Tanco, L. M. and Hilton, A. (2000). Realistic synthesis of novel human movements from a database of motion capture examples. In *Workshop on Human Motion*, pages 137–142. IEEE Press. 32

Tilmanne, J., Moinet, A., and Dutoit, T. (2012). Stylistic gait synthesis based on hidden markov models. *EURASIP Journal on advances in signal processing*, 1:1–14. 33

Tipping, M. and Bishop, C. (1999a). Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482. 72

Tipping, M. E. and Bishop, C. M. (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622. 72

Tolani, D., Goswami, A., and Badler, N. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388. 23

Torresani, L., Hackney, P., and Bregler, C. (2007). Learning motion style synthesis from perceptual observations. In *Advances in Neural Information Processing Systems*, pages 1393–1400. 40

Tsang, W., Singh, K., and Fiume, E. (2005). Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 319–328. Eurographics Association. 43, 44

Unuma, M., Anjyo, K., and Takeuchi, R. (1995). Fourier principles for emotion-based human figure animation. In *22nd annual conference on Computer graphics and interactive techniques*, pages 91–96. ACM Press. 26, 28, 41

Unzueta, L., Peinado, M., Boulic, R., and Suescun, Á. (2008). Full-body performance animation with sequential inverse kinematics. *Graphical models*, 70(5):87–104. 37

Urtasun, R., Glardon, P., Boulic, R., Thalmann, D., and Fua, P. (2004). Style-based motion synthesis. *Computer Graphics Forum*, 23(4):799–812. 40

Van Basten, B. and Egges, A. (2012). Motion transplantation techniques: A survey. *IEEE Computer Graphics and Applications*, 32(3):16–23. 28, 43

Van Basten, B., Stuvel, S., and Egges, A. (2011). A hybrid interpolation scheme for footprint-driven walking synthesis. In *Graphics Interface*, pages 9–16. 34, 35, 36, 46

Van Basten, B. J. H., Peeters, P. W. A. M., and Egges, A. (2010). The step space: examplebased footprintdriven motion synthesis. *Computer Animation and Virtual Worlds*, 21(34):433–441. 56

Van de Panne, M. (1997). From footprints to animation. *Computer Graphics Forum*, 16(4):211–224. 35

Van Welbergen, H., Van Basten, B., Egges, A., Ruttkay, Z., and Overmars, M. (2010). Real time animation of virtual humans: a trade-off between naturalness and contro. *Computer Graphics Forum*, 29(8):2530–2554. 22, 30

Verbeek, J. (2004). *Mixture models for clustering and dimension reduction*. PhD thesis, Universiteit van Amsterdam. 73

Vlasic, D., Adelsberger, R., Vannucci, G., Barnwell, J., Gross, M., Matusik, W., and Popović, J. (2007). Practical motion capture in everyday surroundings. *ACM Transactions on Graphics*, 26(3):Article No. 35. 5

Wampler, C. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least- squares methods. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):93–101. 23

Wang, J., Drucker, S. M., Agrawala, M., and Cohen, M. F. (2006). The cartoon animation filte. *ACM Transactions on Graphics*, 25(3):1169–1173. 1, 26

Wang, J. M., Fleet, D. J., and Hertzmann, A. (2007). Multifactor gaussian process models for style-content separation. In *International conference on Machine learning*, pages 975–982. ACM Press. 33

Wang, X. and Verriest, J. P. (1998). A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation. *The journal of visualization and computer animation*, 9(1):33–47. 11

Weise, T., Bouaziz, S., Li, H., and Pauly, M. (2011). Realtime performance-based facial animation. *ACM Transactions on Graphics*, 30(4):Article No. 77. 33, 40

Wheatland, N., Jörg, S., and Zordan, V. (2013). Automatic hand-over animation using principle component analysis. In *Motion on Games*, pages 175–180. ACM Press. 42, 44

Wiley, D. J. and Hahn, J. K. (1997a). Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45. 11

Wiley, D. J. and Hahn, J. K. (1997b). Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45. 28

Winter, D. (1990). *Biomechanics and Motor Control of Human Movement*. Wiley. 25

Witkin, A. and Popovic, Z. (1995). Motion warping. In *22nd annual conference on Computer graphics and interactive techniques*, pages 105–108. 27

Wooten, W. and Hodgins, J. (1996). Animation of human diving. *Computer Graphics Forum*, 15(1):3–13. 25

Wu, C., Medina, J., and Zordan, V. (2008). Simple steps for simply stepping. In *nternational Symposium on Advances in Visual Computing*, pages 97–106. Springer Berlin Heidelberg. 34, 35, 36, 46

Wu, C. C. and Zordan, V. (2010). Goal-directed stepping with momentum control. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 113–118. Eurographics Association. 1

Xsens (accessed 30/5/2014). Motion capture solution. `http://www.xsens.com/`. 4

Yamane, K., Ariki, Y., and Hodgins, J. (2010). Animating non-humanoid characters with human motion data. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 169–178. Eurographics Association. 1

Yamane, K. and Sok, K. W. (2010). Planning and synthesizing superhero motions. In *Motion in Games*, pages 254–265. Springer Berlin Heidelberg. 1

Yang, P. F., Laszlo, J., and Singh, K. (2004). Layered dynamic control for interactive character swimming. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 39–47. Eurographics Association. 25

Ye, Y. and Liu, C. K. (2012). Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics*, 31(4):Article No. 41. 43

Yoon, H. S., Soh, J., Bae, Y. J., and Seung Yang, H. (2001). Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491–1501. 81, 100

Zeltzer, D. (1982). Motor control techniques for figure animation. *IEEE Computer Graphics and Applications*, 2(9):53–59. 23

Zeltzer, D. (1983). Knowledge-based animation. In *ACM SIGGRAPH/SIGART Workshop on Motion*, pages 187–192. 23

Zhu, Y., Ramakrishnan, A. S., Hamann, B., and Neff, M. (2013). A system for automatic animation of piano performances. *Computer Animation and Virtual Worlds*, 24(5):445–457. 43, 44

Zordan, V. B., Celly, B., Chiu, B., and DiLorenzo, P. C. (2004). Breathe easy: model and control of simulated respiration for animation. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 29–37. Eurographics Association. 25

Zordan, V. B., Majkowska, A., Chiu, B., and Fast, M. (2005). Dynamic response for motion capture animation. *ACM Transactions on Graphics*, 24(3):697–701. 25